

# **ISO TC184/SC4/\* 3 N 309 (P 1 )**

\*Complete with EC (for Editing), PMAG, or WG

**Date:** May 14, 1994

**Supercedes SC4/3 N 275 (P 1 )**

## **PRODUCT DATA REPRESENTATION AND EXCHANGE**

**Part:** 205    **Title:** Application Protocol for Mechanical Design Using Surface Representation

Purpose of this document as it relates to the target document is:

- |                                                     |                              |
|-----------------------------------------------------|------------------------------|
| <input checked="" type="checkbox"/> Primary Content | Current Status: <u>Draft</u> |
| <input type="checkbox"/> Issue Discussion           |                              |
| <input type="checkbox"/> Alternate Proposal         |                              |
| <input type="checkbox"/> Partial Content            |                              |

### **ABSTRACT:**

This document specifies the information model for Part 205 (Mechanical Design Surface AP) of ISO 10303. It comprises 3 functional data groups: one for geometrically bounded, one for non-manifold and one for manifold surface models, each of them handling both analytic and parametric (NURBS) geometry.

### **KEYWORDS:**

Application Protocol, Surface Model, Mechanical Design

### **Document Status/Dates (dd/mm/yy)**

Part Documents	Other SC4 Documents
<u>06/90</u> Released	<u>          </u> Released
<u>06/90</u> Project	<u>          </u> Working
<u>05/94</u> Working	<u>          </u> Editorial OK
<u>05/94</u> Technically Complete	<u>          </u> Technically
<u>06/94</u> Editorially Complete	<u>          </u> Complete
<u>06/94</u> ISO Committee Draft	<u>          </u> Approved

**Owner/Editor:** Jochen Haenisch

**Address:** SINTEF,  
P.O.Box 124, Blindern,  
N-0314 Oslo  
Norway

**Alternate:** Per Evensen

**Address:** SINTEF,  
P.O.Box 124, Blindern,  
N-0314 Oslo  
Norway

**Telephone/FAX:** +47 22 06 73 00/73 50

**E-mail:** Jochen.Haenisch@si.sintef.no

**Telephone/FAX:** +47 22 06 73 00/73 50

**E-mail:** Per.Evensen@si.sintef.no

### **Comments to Reader**

This is the CD-version of Part 205. The AIM is based on the DIS-versions of the Integrated Resources. After the completion of the CD ballot it will be updated to conform to the final IS versions of Parts 41-46.



**Contents**

## **Foreword**

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75% of the member bodies casting a vote.

International Standard ISO 10303–205 was prepared by Technical Committee ISO 184, *Industrial automation systems and integration*, Subcommittee 4, *Industrial data and global manufacturing programming languages*.

The preparation of this document has benefitted from the technical contributions of many projects and their sponsoring organizations. The contributions of the following are acknowledged:

- ESPRIT II 2195, CADEX;
- ESPRIT III, 6040, PRODEX;
- ESPRIT III, 6457, InterRob.

This is the first edition of this part of ISO 10303.

Annexes A, B, C, and D are an integral part of this part of ISO 10303. Annexes E, F, G, H, and J are for information only.

This part is one in a series of parts which together form the International Standard ISO 10303, Industrial automation systems and integration – Product data representation and exchange. At the time of publication of this part of ISO 10303, the following parts had been registered for international ballot:

- Part 1 Overview and fundamental principles;
- Part 11 Description methods: The *EXPRESS* language reference manual;
- Part 21 Implementation methods: Clear text encoding of the exchange structure;
- Part 31 Conformance testing methodology and framework: General concepts;
- Part 32 Conformance testing methodology and framework: Requirements on testing laboratories and clients;
- Part 41 Integrated generic resources: Fundamentals of product description and support;

- Part 42 Integrated generic resources: Geometric and topological representation;
- Part 43 Integrated generic resources: Representation structures;
- Part 44 Integrated generic resources: Product structure configuration;
- Part 45 Integrated generic resources: Materials;
- Part 46 Integrated generic resources: Visual presentation;
- Part 47 Integrated generic resources: Shape variation tolerances;
- Part 49 Integrated generic resources: Process structure and properties;
- Part 101 Integrated application resources: Draughting;
- Part 104 Integrated application resources: Finite element analysis;
- Part 201 Application protocol: Explicit draughting;
- Part 202 Application protocol: Associative draughting;
- Part 203 Application protocol: Configuration controlled 3D design of mechanical parts and assemblies;
- Part 207 Application protocol: Sheet metal die planning and design.

The reader may obtain information on the other Parts of ISO 10303 from the ISO Central Secretariat.

## **Introduction**

ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and archiving.

ISO 10303 is organized as a series of parts, each published separately. The parts of this International Standard fall into one of the following series: description methods, integrated resources, application protocols, abstract test suites, implementation forms, and conformance testing. The series are described in ISO 10303-1. This part of ISO 10303 is a member of the application protocols series.

An application protocol provides a scope and a context for the general-purpose constructs defined in the integrated resources class of parts. Since adequate conformance testing can only be based on a valid set of requirements, implementations can only be tested in terms of an application protocol.

This part of ISO 10303 specifies an application protocol (AP) for the representation and exchange of digital surface models as they are typically used in mechanical computer-aided design. The specified model is meant to ease access to surface design data from disciplines that rely on design data such as engineering, simulation, visualisation, manufacturing. The transfer and archiving of surface models in this environment, at different stages of the design and engineering process, requires the following to be maintained:

- completeness of mapping between application systems;
- correctness of semantics of the representation;
- accuracy of relationships between model entities and instances.

This part of ISO 10303 supports data access by file exchange and by a procedural interface to data repositories.

Three alternative types of surface models are specified in this part of ISO 10303:

- models without explicit topology;
- models with manifold topology;
- models with non-manifold topology.

Each of these representations supports both elementary and B-spline curves and surfaces.

A surface model describes a certain functionality for a version of a product. It may be linked to other representations of the same product. Surface models may also be assembled.

In this part of ISO 10303 surface models may be assigned visualisation attributes such as line font and colour.

The individual elements of a surface model, as well as the surface models themselves, may be grouped or, for visualisation purposes, put on layers. Most types of elements of a surface model

may be assigned user-defined names.

Measures such as coordinates are assigned SI-units. Only angles may be of converted units such as degrees. Units shall be assigned globally, for instance to entire surface models.

This application protocol defines the context, scope, and information requirements for mechanical design using surface models and specifies the integrated resources necessary to satisfy these requirements.

Application protocols provide the basis for developing implementations of ISO 10303 and abstract test suites for the conformance testing of AP implementations.

Clause 1 defines the scope of the application protocol and summarizes the functionality and data covered by the AP. An application activity model that is the basis for the definition of the scope is provided in annex E. The information requirements of the application are specified in clause 4 using terminology appropriate to the application. A graphical representation of the information requirements, referred to as the application reference model, is given in annex F.

Resource constructs are interpreted to meet the information requirements. This interpretation produces the application interpreted model (AIM). This interpretation, given in 5.1, shows the correspondence between the information requirements and the AIM. The short listing of the AIM specifies the interface to the integrated resources and is given in 5.2. Note that the definitions and *EXPRESS* provided in the the integrated resources for constructs used in the AIM may include select list items and subtypes which are not imported into the AIM. The expanded listing given in annex A contains the complete *EXPRESS* for the AIM without annotation. A graphical representation of the AIM is given in annex G. Additional requirements for specific implementation methods are given in annex D.

# Industrial automation systems and integration - Product data representation and exchange - Part 205: Application protocol: Mechanical design using surface representation

## 1 Scope

This part of ISO 10303 specifies the integrated resources necessary for the scope and information requirements for mechanical design using surface representation.

NOTE – The application activity model in annex E provides a graphical representation of the processes and information flows which are the basis for the definition of the scope of this part of ISO 10303.

The following are within the scope of this part of ISO 10303:

- representation of surface design data, as utilized in computer aided mechanical design, for the purpose of data exchange;
- styling, design and engineering activities;
- activities utilising CAD-modelling techniques;
- representations of shapes with surface models;
- controlled use of only specific topological representations within a surface model as a means of minimizing the number of conversions between alternative surface model representations;
- explicit representations of curve and surface topology;
- representation of the fact of whether a set of surfaces is closed in order to maintain compatibility with boundary representation (B-rep) models;
- implicit representations of point, curve and surface topology when explicit topology is not present; i.e. the representation of trimming, composition, on-curve relationships, on-surface relationships, intersection, offset;
- representation of those non-manifolds in which faces and edges may be shared;

NOTE 1 – This item has been included to satisfy especially some of the needs of finite element analysis applications.

- non-uniform rational B-spline (NURBS) geometry;

## **ISO/CD 10303-205**

- representation of elementary geometry, i.e. line, circle, ellipse, hyperbola, parabola, plane surface, cylindrical surface, conical surface, toroidal surface, and spherical surface;
- representation of simple swept geometry, i.e. linear extrusion and revolution;
- geometry resulting from sweeping operations along arbitrary curves, but not the representation of the operations themselves;
- handling of constructive geometry;
- information on recursive translations, rotations, mirroring and uniform scaling of geometric entities and surface models;
- visual presentation of surface models;
- representation of display properties for points, curves, and surfaces;
- representation of text that is located in the plane of the screen image;
- representation of layers as used for grouping of parts of surface models for visual presentation;
- representation of product structure including assembly information;
- representation of user-defined names of objects;
- stable representation of dimensions and coordinates;
- representation of globally applicable units;
- measurement of length and angle.

The following are outside the scope of this part of ISO 10303:

- surface model representations that are the result of processing the product design information for tasks, such as analysis or numerically controlled production;
- non-shape information, such as configuration control, including information describing physical properties, such as material and surface conditions;
- finite element analysis information, such as boundary conditions, loads, analysis results;
- procedurally defined curves and surfaces except for offset curves and surfaces;
- self-intersecting geometry;
- 2D geometry except for 2D curves used in the definition of curves within the parameter space of a surface;

- representation of B-rep solids;
- representation of CSG solids;
- presentation features, such as character fonts and symbols except for literal text symbols;
- production form features, such as holes, slots, etc. ;
- dimensions and tolerances;
- individual assignment of units to single measures with units.

NOTE 2 – Figure ?? indicates where in mechanical design and engineering the scope of this part of ISO 10303 is located. Along the horizontal axis, disciplines are listed that are involved in the design of mechanical products. The vertical axis represents modelling techniques. This part satisfies the requirements of the styling and design of mechanical parts. Both sculptured and elementary surface geometry and a set of presentation attributes are in scope. The figure illustrates the close relationship between surface, solid, and wireframe representations as far as mechanical part design, as shown in the centre column, is concerned. This close relationship is maintained by this part of ISO 10303; the closed shell entity provides the interface to solid models, the different curve entities can be used for interfacing to wireframe models. Subsequent manufacturing processes such as production planning, analysis, and simulation may take advantage of the surface representations that are conformant to this part of ISO 10303.

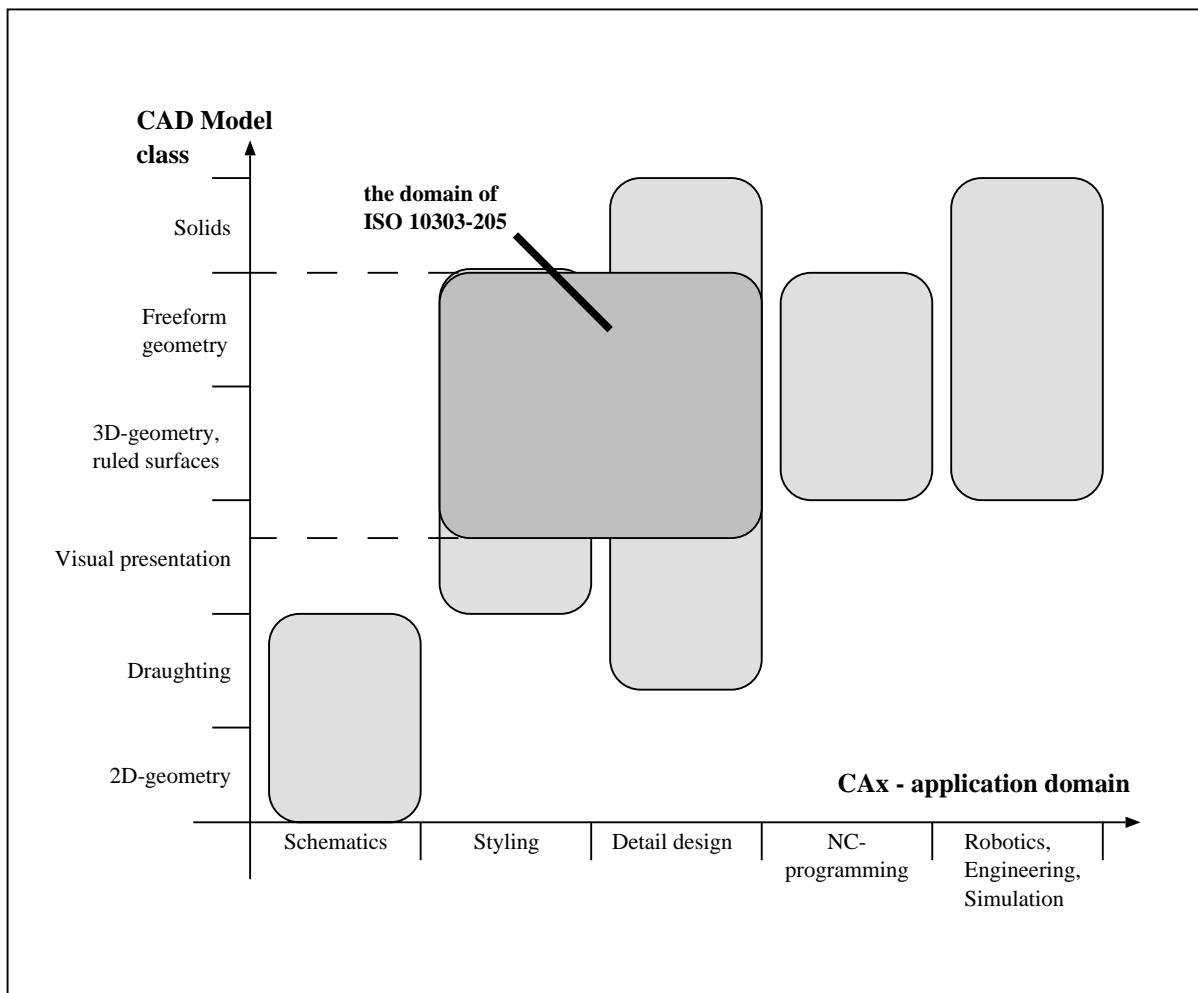


Figure 1 – The scope of this part of ISO 10303 and its relation to different mechanical design and manufacturing activities

## 2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO 10303. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

- |                         |                                                                                                                                                                                                              |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ISO 10303-1</b>      | <i>Industrial automation systems and integration – Product data representation and exchange – Part 1 : Overview and fundamental principles.</i>                                                              |
| <b>ISO 10303-11</b>     | <i>Industrial automation systems and integration – Product data representation and exchange – Part 11 : Description methods: The EXPRESS language reference manual.</i>                                      |
| <b>ISO 10303-21</b>     | <i>Industrial automation systems and integration – Product data representation and exchange – Part 21 : Clear text encoding of the exchange structure.</i>                                                   |
| <b>ISO/CD 10303-22</b>  | <i>Industrial automation systems and integration – Product data representation and exchange – Part 22 : Standard data access interface specification.</i>                                                    |
| <b>ISO 10303-31</b>     | <i>Industrial automation systems and integration – Product data representation and exchange – Part 31 : Conformance testing methodology and framework: General concepts.</i>                                 |
| <b>ISO/DIS 10303-32</b> | <i>Industrial automation systems and integration – Product data representation and exchange – Part 32 : Conformance testing methodology and framework: Requirements on testing laboratories and clients.</i> |
| <b>ISO 10303-41</b>     | <i>Industrial automation systems and integration – Product data representation and exchange – Part 41 : Integrated generic resources: Fundamentals of product description and support.</i>                   |
| <b>ISO 10303-42</b>     | <i>Industrial automation systems and integration – Product data representation and exchange – Part 42 : Integrated generic resources: Geometric and topological representation.</i>                          |
| <b>ISO 10303-43</b>     | <i>Industrial automation systems and integration – Product data representation and exchange – Part 43 : Integrated generic resources: Representation structures.</i>                                         |

**ISO/CD 10303-205**

- ISO 10303-46** *Industrial automation systems and integration – Product data representation and exchange – Part 46 : Integrated generic resources: Visual presentation.*
- ISO/WD 10303-1205** *Industrial automation systems and integration – Product data representation and exchange – Part 1205 : Abstract test suite: Mechanical design using surface representation.*
- ISO TC184/SC4/WG4 N603c** *Application interpreted construct: Topology bounded surface*
- ISO TC184/SC4/WG4 N607c** *Application interpreted construct: Geometrically bounded surface shape representation*
- ISO TC184/SC4/WG4 N608c** *Application interpreted construct: Non-manifold surface shape representation*
- ISO TC184/SC4/WG4 N609d** *Application interpreted construct: Manifold surface shape representation*
- ISO TC184/SC4/WG4 N618c** *Application interpreted construct: Mechanical design context*
- ISO TC184/SC4/WG4 N621** *Application interpreted construct: Mechanical design presentation*

### 3 Definitions and abbreviations

#### 3.1 Terms defined in ISO 10303-1

This part of ISO 10303 makes use of the following terms defined in ISO 10303-1:

- **application:**
- **application activity model (AAM):**
- **application context:**
- **application interpreted model (AIM):**
- **application protocol (AP):**
- **application reference model (ARM):**
- **assembly:**
- **component:**
- **conformance testing:**
- **context:**
- **data:**
- **data exchange:**
- **generic resource:**
- **implementation method:**
- **information:**
- **integrated resource (IR):**
- **interpretation:**
- **model:**
- **PICS:**
- **PIXIT:**

- **product:**
- **product data:**
- **resource construct:**
- **structure:**
- **test purpose:**
- **unit of functionality (UoF):**

### 3.2 Terms defined in ISO 10303-31

This part of ISO 10303 makes use of the following terms defined in ISO 10303-31:

- **conformance assessment process:**
- **conformance class:**

### 3.3 Terms defined in ISO 10303-42

This part of ISO 10303 makes use of the following terms defined in ISO 10303-42:

- **arcwise connected:**
- **axi-symmetric:**
- **bounds:**
- **boundary:**
- **closed curve:**
- **closed surface:**
- **completion of a topological entity:**
- **connected:**
- **connected component:**
- **curve:**
- **cycle:**
- **dimensionality:**

- **domain:**
- **Euler equations:**
- **extent:**
- **finite:**
- **genus of a graph:**
- **genus of a surface:**
- **geometric coordinate system:**
- **graph:**
- **handle:**
- **homeomorphic:**
- **inside:**
- **interior:**
- **list:**
- ***d*-manifold with boundary:**
- **open curve:**
- **open surface:**
- **orientable:**
- **overlap:**
- **parameter range:**
- **parameter space:**
- **placement coordinate system:**
- **self-intersect:**
- **self-loop:**
- **set:**

## ISO/CD 10303-205

- **space dimensionality:**
- **surface:**
- **topological sense:**

### 3.4 Terms defined in ISO 10303-43

This part of ISO 10303 makes use of the following terms defined in ISO 10303-43:

- **coordinate space:**
- **geometrically founded:**
- **geometrically related:**

### 3.5 Terms defined in ISO 10303-46

This part of ISO 10303 makes use of the following terms defined in ISO 10303-46:

- **externally defined:**
- **physical state variable:**
- **picture:**
- **predefined items:**
- **presentation:**
- **presentation information:**
- **product shape:**
- **realistic presentation of properties:**
- **staircase function:**
- **symbol:**
- **symbolic presentation of properties:**
- **synthetic camera model:**
- **visualisation:**

## 3.6 Other definitions

For the purposes of this part of ISO 10303, the following definitions apply:

**3.6.1 2-manifold:** a topology that is suitable for the development of solid models.

**3.6.2 application interpreted construct (AIC):** a logical grouping of concepts that is shared by two or more AIMs. An application interpreted construct may represent a unit of functionality defined in an application reference model.

**3.6.3 application object:** an atomic element of an application reference model that defines a unique concept and contains attributes specifying its data elements.

**3.6.4 boundary representation (B-rep):** a solid model with boundaries that are represented by a set of surfaces with certain topological constraints.

**3.6.5 constructive solid geometry (CSG):** a solid defined by a collection of primitive solids, combined using regularised boolean operations. A sphere is an example of a primitive solid. Intersection, union and difference are examples of regular boolean operations.

**3.6.6 finite element analysis (FEA):** a mathematical method for calculating stress and deformation of constructions under load.

**3.6.7 functional data group (FDG):** an alternative representation of a surface model. The functional data groups of surface models differ mainly in topological complexity.

**3.6.8 geometrically bounded surface model:** a representation of a surface model not using any topological entities in which geometry instead is bounded by geometrical entities or parameter values.

**3.6.9 initial graphics exchange specification (IGES):** an ANSI-standard for the exchange of product model data.

**3.6.10 manifold surface model:** a representation of a surface model using topological entities in which the topological relationships are 2-manifold.

**3.6.11 non-manifold surface model:** a representation of a surface model using topological entities in which the topological relationships may be non-manifold.

**3.6.12 numerical control (NC):** a control mechanism for switching and positioning of machining tools.

**3.6.13 standard d'échange et de transfert (SET):** a French national standard for the exchange of product model data.

**3.6.14 Verband der Automobilindustrie - Flaechenschnittstelle (VDAFS):** a specification for the exchange of surface model shape data recommended by the German car industry association.

**3.6.15 Verband der Automobilindustrie - IGES subset (VDAIS):** an IGES subset recommended by the German car industry association.

### **3.7 Abbreviations**

For the purposes of this part of ISO 10303, the following abbreviations apply:

AAM	Application Activity Model
AIC	Application Interpreted Construct
AIM	Application Interpreted Model
AP	Application Protocol
ARM	Application Reference Model
B-rep	Boundary Representation
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CIM	Computer Integrated Manufacturing
CSG	Constructive Solid Geometry
FDG	Functional Data Group
FEA	Finite Element Analysis
ICAM	Integrated Computer-Aided Manufacturing
IDEF0	ICAM Definition Language 0
IGES	Initial Graphics Exchange Specification
mm	millimetre
NIAM	Nijssen's Information Analysis Method
NC	Numerical Control
NURBS	Non Uniform Rational B-Spline
PICS	Protocol Information and Conformance Statement

SI	International System of Units
UoF	Unit of Functionality
VDAFS	Verband der Automobilindustrie - Flaechenschnittstelle
VDAIS	Verband der Automobilindustrie - IGES subset
WSN	Wirth Syntax Notation

## 4 Information requirements

This clause specifies the information required for mechanical design using surface representation. The information requirements are specified as a set of units of functionality, application objects, and application assertions. These assertions pertain to individual application objects and to relationships between application objects. The information requirements are defined using the terminology of the subject area of this application protocol.

### NOTES

- 1 – A graphical representation of the information requirements is given in annex F.
- 2 – The information requirements correspond to those of the activities identified as being in the scope of this application protocol in annex E.
- 3 – The mapping table is specified in 5.1 which shows how the information requirements are met using the integrated resources of this International Standard. The use of the integrated resources introduces additional requirements which are common to application protocols.

### The functional data groups of this part

This part of ISO 10303 specifies three shape representations for surface models which are organised into three different functional data groups (FDG):

- Functional Data Group 1 (FDG1) – geometrically bounded surface model;
- Functional Data Group 2 (FDG2) – non-manifold surface model;
- Functional Data Group 3 (FDG3) – manifold surface model.

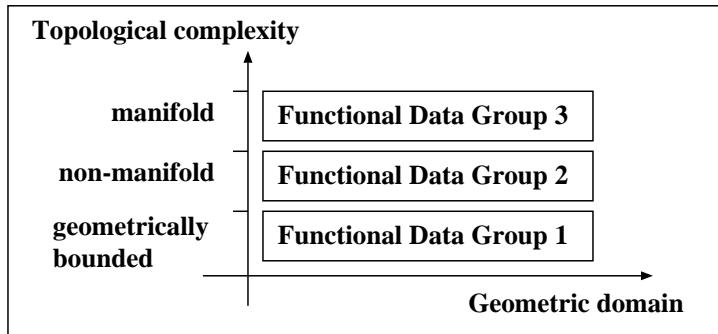
These functional data groups are the basis for the conformance classes as they are specified in clause 6. Each functional data group may require visual presentation information. The ARM in Annex E is structured to show contents and relationships of the functional data groups.

Some surface model representations have contradictory requirements, such as support of manifoldness and non-manifoldness. Other requirements produce almost redundant surface model representations, such as a surface model with explicit topology and one with implicit topology. In order to avoid such contradictions and redundancies within one application, only one of the three alternative surface model representations shall be used at a time. The conformance classes of this part of ISO 10303 ensure this exclusive usage.

The AIM of this part comprises the specifications of all three representations. The user of this part may select the most appropriate functional data group for a given task. The three options are formally specified in ?? by the rule **alternative\_surface\_shape\_representations** for entity **shape\_representation**.

### Functional data group usage

The degree of support for surface topology distinguishes the three functional data groups. The relationships between them are illustrated in figure ?.?. All three functional data groups support the same geometrical features; support for geometry is indicated by the horizontal axis.



**Figure 2 – The three functional data groups represent different topological categories of surface models.**

Topologically the most restricted surface model representation is the manifold one followed by non-manifold and geometrically bounded.

For all three functional data groups is required that:

- product structure information shall be included;
- global units shall be included;
- visual presentation attributes may be assigned;
- objects may be grouped;
- user-defined names may be applied.

#### **Geometrically bounded surface model**

Functional data group 1 provides support for surface models that do not include topological objects such as vertex, edge, face. The same basic geometric functionality is included as in the other two surface model representations; both elementary and free-form curves and surfaces. Information concerning trimming, composition, intersection and offset of curves and surfaces is present. There are, however, limitations on the composition of surfaces. The resulting surface is composed of a rectangular matrix of patches each of which must be four-sided. On-curve and on-surface information is included. Both single geometric objects such as curves and surfaces and complete surface models may be replicated. The information domain of geometrically bounded surface models includes wireframe models of corresponding functionality.

#### **Non-manifold surface model**

Functional data group 2 represents surface models which may have limited non-manifold topology capabilities.

**EXAMPLE 1 –** FEA-applications may be regarded typical users of the non-manifold surface representation.

This level allows

**Table 1 – The use of units of functionality in functional data groups**

<b>UoF/FDG</b>	<b>Geometrically bounded (FDG1)</b>	<b>Non-manifold (FDG2)</b>	<b>Manifold (FDG3)</b>
geometrically_bounded_surface_model_shape non_manifold_surface_model_shape manifold_surface_model_shape surface_basic_visual_presentation (optional) product_structure grouping	X  X  X  X	X  X  X  X	X  X  X  X

- for two connected face sets to have overlapping faces provided that the face boundaries are identical;
- for more than two faces to share one edge.

No other non-manifold features are supported. This surface model representation requires topological objects. The use of topological objects enables explicit modelling of relationships for improved representation of connectivity. Information concerning trimming and composition of curves and surfaces is provided by utilizing topological objects such as vertex, edge and face. Both open and closed surface models may be represented. The non-manifold surface representation has the same capabilities of the manifold surface representation except that the semantics of manifoldness are not explicitly provided.

#### **Manifold surface model**

Functional data group 3 represents manifold surface models in the sense of 2-manifolds. This functional data group requires topological objects. As for the non-manifold surface representation, all outer-bound information can be maintained including the information that a surface model is open or closed. The topological constraints on manifold surface models are, however, more restrictive than those on non-manifold surface models. In particular

- may faces be shared, but shall not overlap;
- shall an edge be shared by at maximum two faces.

EXAMPLE 2 – These restrictions imply that a manifold surface representation can convey the fact that a surface model is both closed and a manifold. These are requirements that a B-rep solid model needs to fulfill. The manifold surface model representation can, therefore, be regarded as the one of the three surface representations of this part of ISO 10303 that is closest to the B-rep representation.

## 4.1 Units of functionality

This subclause specifies the units of functionality for the mechanical design using surface representation application protocol. This part of ISO 10303 specifies the following units of functionality:

- `geometrically_boundedsurface_model_shape`;
- `grouping`;
- `manifold_surface_model_shape`;
- `non_manifold_surface_model_shape`;
- `product_structure`;
- `surface_basic_visual_presentation`.

The units of functionality and a description of the functions that each UoF supports are given below. The application objects included in the UoF's are defined in 4.2.

Table ?? specifies the correlation between functional data groups and UoFs, i.e. the use of the UoFs listed above in the three functional data groups of this part.

### 4.1.1 `geometrically_boundedsurface_model_shape`

The `geometrically_boundedsurface_model_shape` UoF represents a surface model that does not contain any topological objects such as vertex, edge, loop, and face. It meets the requirements of applications that need not support connectivity relationships explicitly. This UoF provides geometrically the same basic functionality as the other surface models specified in this part of ISO 10303. So both elementary and free-form curves and surfaces may be represented. Information concerning trimming, composition, intersection and offset curves and surfaces is included. Composition of surfaces is restricted to include only composite surfaces that are composed of rectangular matrices of patches, each of which shall be four-sided. Geometry may be represented in the parameter space of a curve or a surface by using application objects such as `Point_on_curve` and `Parametric_curve`. Both single geometric objects, such as curves and surfaces, and composed objects, such as surface models may be replicated. This UoF does not require surfaces to be part of a model. A model may consist of a set of curves only.

The following application objects are used by `geometrically_boundedsurface_model_shape` UoF:

- `Axis_placement`;
- `B_spline_curve`;
- `B_spline_surface`;
- `Bounded_curve`;
- `Bounded_surface`;

## **ISO/CD 10303-205**

- Cartesian\_point;
- Circle;
- Composite\_curve;
- Composite\_curve\_on\_surface;
- Conical\_surface;
- Curve;
- Curve\_2d;
- Curve\_3d;
- Curve\_bounded\_surface;
- Curve\_on\_surface;
- Curve\_replica;
- Cylindrical\_surface;
- Degenerated\_parametric\_curve;
- Elementary\_surface;
- Ellipse;
- Geometrically\_bounded\_surface\_model;
- Geometric\_element;
- Geometry\_replica;
- Hyperbola;
- Intersection\_curve;
- Line;
- Offset\_curve;
- Offset\_surface;
- Parabola;

- Parametric\_curve;
- Plane;
- Point;
- Point\_on\_curve;
- Point\_on\_surface;
- Polyline;
- Rectangular\_composite\_surface;
- Rectangular\_trimmed\_surface;
- Seam\_curve;
- Spherical\_surface;
- Surface;
- Surface\_model;
- Surface\_model\_replica;
- Surface\_of\_linear\_extrusion;
- Surface\_of\_revolution;
- Surface\_replica;
- Swept\_surface;
- Toroidal\_surface;
- Transformation;
- Trimmed\_curve;
- Unbounded\_curve.

#### 4.1.2 grouping

The grouping UoF contains information on organisational structures defined within a surface model. It includes the user capabilities to collect geometric, topological, and surface model

objects for a specific discipline or viewpoint. This part of ISO 10303 includes also the ability of grouping for the purpose of visual presentation. This is represented by application object Layer that is specified in ??.

The following application object is used by grouping UoF:

- Group.

#### **4.1.3 manifold\_surface\_model\_shape**

The manifold\_surface\_model\_shape UoF represents manifold surface models that meet the requirements of 2-manifoldness. This UoF requires topology to define outer-bound information. The outer-bound information may indicate that a surface model is closed or open. This UoF provides the same basic geometric functionality as the geometrically\_bounded\_surface\_model\_shape UoF in that both elementary and free-form curves and surfaces may be represented. Information concerning trimming and composition of curves and surfaces is provided by using topological objects such as vertex, edge, loop, and face. Intersection and offset functionality is provided. Geometry may be represented in the parameter space of a curve or a surface by using application objects such as Point\_on\_curve and Parametric\_curve. Both single geometric objects, such as curves and surfaces, and complete surface models may be replicated. The information domain of UoF includes the basic geometry and topology of wireframe models of corresponding functionality.

The following application objects are used by manifold\_surface\_model\_shape UoF:

- Axis\_placement;
- B\_spline\_curve;
- B\_spline\_surface;
- Bounded\_curve;
- Bounded\_surface;
- Cartesian\_point;
- Circle;
- Closed\_shell;
- Conical\_surface;
- Curve;
- Curve\_2d;
- Curve\_3d;

- Curve\_on\_surface;
- Curve\_replica;
- Cylindrical\_surface;
- Degenerated\_parametric\_curve;
- Edge;
- Elementary\_surface;
- Ellipse;
- Face;
- Face\_set;
- Geometric\_element;
- Geometry\_replica;
- Hyperbola;
- Intersection\_curve;
- Line;
- Loop;
- Manifold\_surface\_model;
- Offset\_curve;
- Offset\_surface;
- Open\_shell;
- Parabola;
- Parametric\_curve;
- Plane;
- Point;
- Point\_on\_curve;

- Point\_on\_surface;
- Polyline;
- Seam\_curve;
- Shell;
- Spherical\_surface;
- Surface;
- Surface\_model;
- Surface\_model\_replica;
- Surface\_of\_linear\_extrusion;
- Surface\_of\_revolution;
- Surface\_replica;
- Swept\_surface;
- Topological\_element;
- Topology\_surface\_model;
- Toroidal\_surface;
- Transformation;
- Unbounded\_curve;
- Vertex.

#### **4.1.4 non\_manifold\_surface\_model\_shape**

The non\_manifold\_surface\_model\_shape UoF represents surface models with non-manifold characteristics. This UoF specifically supports shape representation requirements from FEA-applications. This UoF allows two connected face sets to share one face and two or more faces to share one edge. No additional non-manifold features are supported. Information may be attached to connected face sets to identify them as open or closed. The use of topological objects is required. This UoF provides the same basic geometric functionality as the geometrically\_bounded\_surface\_model\_shape UoF. So both elementary and free-form curves and surfaces are supported. Information concerning trimming and composition of curves and surfaces is included

by providing topological objects such as vertex, edge, loop, and face. Intersection and offset functionality is included. Geometry may be represented in the parameter space of a curve or a surface by using application objects such as Point\_on\_curve and Parametric\_curve. Both single geometric objects, such as curves and surfaces, and complete surface models may be replicated. The information domain of UoF includes the basic geometry and topology of wireframe models of corresponding functionality.

The following application objects are used by non\_manifold\_surface\_model\_shape UoF:

- Axis\_placement;
- B\_spline\_curve;
- B\_spline\_surface;
- Bounded\_curve;
- Bounded\_surface;
- Cartesian\_point;
- Circle;
- Closed\_shell;
- Conical\_surface;
- Curve;
- Curve\_2d;
- Curve\_3d;
- Curve\_on\_surface;
- Curve\_replica;
- Cylindrical\_surface;
- Degenerated\_parametric\_curve;
- Edge;
- Elementary\_surface;
- Ellipse;
- Face;

## **ISO/CD 10303-205**

- Face\_set;
- Geometric\_element;
- Geometry\_replica;
- Hyperbola;
- Intersection\_curve;
- Line;
- Loop;
- Non\_manifold\_surface\_model;
- Offset\_curve;
- Offset\_surface;
- Open\_shell;
- Parabola;
- Parametric\_curve;
- Plane;
- Point;
- Point\_on\_curve;
- Point\_on\_surface;
- Polyline;
- Seam\_curve;
- Shell;
- Spherical\_surface;
- Surface;
- Surface\_model;
- Surface\_model\_replica;

- Surface\_of\_linear\_extrusion;
- Surface\_of\_revolution;
- Surface\_replica;
- Swept\_surface;
- Topological\_element;
- Topology\_surface\_model;
- Toroidal\_surface;
- Transformation;
- Unbounded\_curve;
- Vertex.

#### **4.1.5 product\_structure**

The product\_structure UoF contains application objects that are required for product identification, for associating a shape in a mechanical design context, and for defining product assemblies. The identification of parts and of multiple definitions of parts are supported, as well as the versioning of parts. Assemblies may consist of both sub-assemblies and individual parts. Individual parts are represented by surface models. Assemblies define specific geometric relationships between parts, between parts and assemblies, and between assemblies. These relationships are given by the following properties:

- a reference from one assembly to another assembly or to a part;
- a geometrical relationship which may be described with a transformation matrix and which allows at least translation and rotation. Mirroring and scaling may be required;
- a name which should be unique within a product.

This UoF has the capability of associating product shape data from shape representations with different topological dimensionality, i.e. solid models and surface models. The product structure information is provided independently of product shape.

The following application objects are used by the product\_structure UoF:

- Assembly;
- Part;
- Product;

- Shape\_representation;
- Transformation.

#### **4.1.6 surface\_basic\_visual\_presentation**

The `surface_basic_visual_presentation` UoF includes application objects for the visual presentation of surface models. These objects enable the user to specify the appearance of:

- points;
- curves;
- surfaces.

Various attributes, such as point marker type, curve font, surface rendering, may be set. Appearance may be assigned to geometric objects, and also to topological objects such as vertex, edge, and face, and to geometric models. Objects that do not have appearance attributes assigned, shall be invisible. Annotation text is placed into the plane of a screen image without any reference to other presented objects. A layer mechanism is provided for display purposes. Layers may contain geometric objects, some of the topological objects, and complete geometric models. An object instance may belong to several layers.

The following application objects are used by the `surface_basic_visual_presentation` UoF:

- 3D\_projection;
- Annotation\_text;
- Curve\_appearance;
- Layer;
- Point\_appearance;
- Presentation\_appearance;
- Screen\_image;
- Surface\_appearance.

## **4.2 Application objects**

This subclause specifies the application objects for the mechanical design using surface representation application protocol. Each application object is an atomic element that embodies a unique application concept and contains attributes specifying the data elements of the object. The application objects and their definitions are given below.

### **4.2.1 3D\_projection**

A 3D\_projection is a picture of a 3-dimensional shape. The picture is the result of a mapping by a camera model.

### **4.2.2 Annotation\_text**

An Annotation\_text is any arbitrary text that is placed into a Screen\_image. The text is not associated with any of the objects presented in the Screen\_image.

NOTE 1 – The requirement for association between Annotation\_text and any Geometric\_element or Topological\_element has been relaxed as it currently can not be mapped using ISO 10303-46.

### **4.2.3 Assembly**

An Assembly is a collection of Parts that is assigned a location and, optionally associated with an identifier. The location shall be defined by a Transformation. The identifier may be specified by a user defined name. The data associated with an Assembly are the following:

- coordinate\_system;
- user\_defined\_name.

#### **4.2.3.1 coordinate\_system**

The coordinate\_system specifies the distinct coordinate space for the Assembly, spatially unrelated to other coordinate spaces. Other coordinate spaces may be related to the one of the Assembly by means of transformations.

#### **4.2.3.2 user\_defined\_name**

The user\_defined\_name specifies the word, or group of words, by which the Assembly is referred to.

### **4.2.4 Axis\_placement**

An Axis\_placement is a location and orientation used for the definition of geometric objects. It may be defined in terms of a locating point and an axis direction, and it may be used in the definition of axi-symmetric elements, e.g. surface of revolution. It may also be defined in terms of a point and two orthogonal axes, and it may be used to locate and orientate a non-axi-symmetric element in space, e.g. conic curves and elementary surfaces.

### **4.2.5 B\_spline\_curve**

A B\_spline\_curve is a Bounded\_curve that is used to represent a wide variety of free form curves. With appropriate attribute values, it is capable of representing single span or spline curves of

explicite polynomial, rational Bezier, or B-spline types. A B\_spline\_curve shall not self-intersect.

#### **4.2.6 B\_spline\_surface**

A B\_spline\_surface is a Bounded\_surface that is used to represent a wide variety of free form surfaces. With appropriate attribute values it is capable of representing single span or spline surfaces of explicite polynomial, rational Bézier or B-spline types. A B\_spline\_surface shall not self-intersect.

#### **4.2.7 Bounded\_curve**

A Bounded\_curve is a collection of curve types with natural or artificial start and end points, i.e. curves with a finite arc length. Trimmed\_curve, Composite\_curve, Polyline, and B\_spline\_curve are all Bounded\_curves. All of these subtypes are valid in the context of Geometrically\_boundedsurface\_models. Not all of them are valid in the contexts of Non\_manifold\_surface\_models and Manifold\_surface\_models. Bounding of curves is required for this part of ISO 10303 and may be done by using this application object or alternatively by using topological constructs such as Vertex and Edge. A Bounded\_curve shall not by itself be instantiated.

#### **4.2.8 Bounded\_surface**

A Bounded\_surface is a collection of surface types with a natural or artificial outer boundary, and optionally inner boundaries. Rectangular\_trimmed\_surface, Rectangular\_composite\_surface, curve\_boundedsurface, and B\_spline\_surface are Bounded\_surfaces. All of these subtypes are valid in the context of Geometrically\_boundedsurface\_models. Not all of these are valid in the contexts of Non\_manifold\_surface\_models and Manifold\_surface\_models. Bounding of surfaces is required for this part of ISO 10303 and may be done by using this application object or alternatively by using topological constructs such as Edge and Face. A Bounded\_surface shall not by itself be instantiated.

#### **4.2.9 Cartesian\_point**

A Cartesian\_point is a Point defined in a two or three dimensional rectangular cartesian coordinate system.

#### **4.2.10 Circle**

A Circle is a planar closed curve at a constant distance from a centre point. A Circle may be trimmed to a circular arc by using a Trimmed\_curve object or the topological objects Vertex and Edge.

#### **4.2.11 Closed\_shell**

A Closed\_shell is a collection of one or more Faces. Distinct Faces of the Shell shall not intersect, but they may share Edges. A Closed\_shell may serve as a bound for a region in 3D and, thus, it divides a 3D coordinate space into two connected regions, one finite and the other infinite. The topological normal of the Closed\_shell is defined to be directed from the finite to the infinite.

#### **4.2.12 Composite\_curve**

A Composite\_curve is a Bounded\_curve and is a collection of Bounded\_curves joined end to end. A Composite\_curve shall not self-intersect.

#### **4.2.13 Composite\_curve\_on\_surface**

A Composite\_curve\_on\_surface is a Curve\_on\_surface that is a collection of Curve\_on\_surfaces joined end to end.

#### **4.2.14 Conical\_surface**

A Conical\_surface is a Surface which could be produced by revolving a straight line in 3D space about any intersecting straight line. The angle between the axis and the revolved line shall be between and not equal to 0 and 90 degrees. The Surface shall always be trimmed to a finite extension either by assigning it to a Face or by specifying parameter values using the objects Rectangular\_trimmed\_surface or Curve\_bounded\_surface.

#### **4.2.15 Curve**

A Curve is the path of a point moving within a coordinate space. Curve\_2D, Curve\_3D, and Curve\_replica are subtypes of Curve. Curves are used in the definition of other geometric objects such as Point and Surface. A Curve shall not by itself be instantiated.

#### **4.2.16 Curve\_2D**

A Curve\_2D is a Curve that is defined in 2D space, i.e. the curve is located within a 2D coordinate system and, if it is described by points, these points are given by only 2 coordinates. A Curve\_2D shall not by itself be instantiated.

#### **4.2.17 Curve\_3D**

A Curve\_3D is a Curve that is defined in 3D space, i.e. the curve is located within a 3D coordinate system and, if it is described by points, these points are given by 3 coordinates. A Curve\_3D shall not by itself be instantiated.

## 4.2.18 Curve\_appearance

A Curve\_appearance specifies the visual appearance of a curve.

The data associated with a Curve\_appearance are the following:

- colour;
- curve\_font;
- curve\_width.

### 4.2.18.1 colour

The colour is optional and, if present, defines the basic appearance property with respect to the light reflectance by a curve. Colour shall be specified by intensity values for red, green, and blue.

### 4.2.18.2 curve\_font

The curve\_font is optional and, if present, describes the lengths of the visible and invisible segments of curves.

### 4.2.18.3 curve\_width

The curve\_width is optional and, if present, sets the width of curves.

## 4.2.19 Curve\_bounded\_surface

A Curve\_bounded\_surface is a Bounded\_surface that is defined by one or more Composite\_curve\_on\_surfaces. One of these curves may be the outer boundary; any number of inner boundaries are permitted. A Curve\_bounded\_surface shall be trimmed by Parametric\_curves only.

## 4.2.20 Curve\_on\_surface

A Curve\_on\_surface is a Curve in the parametric space of a Surface. Parametric\_curve, Seam\_curve, Composite\_curve\_on\_surface are Curves of type Curve\_on\_surface. Composite\_curve\_on\_surfaces shall only be a legal Curve\_on\_surfaces in the context of Geometrically\_boundedsurface\_models. A Curve\_on\_surface shall not by itself be instantiated.

## 4.2.21 Curve\_replica

A Curve\_replica is the replica of an existing Curve\_3d at a different location and that is itself a Curve\_3d. The Transformation between the original and the Curve\_replica may contain scaling. The Curve\_replica is one object even if the original Curve\_3d is a collection of objects. The Points and Curves of the Curve\_replica are not identified and may not be modified as such.

## 4.2.22 Cylindrical\_surface

A Cylindrical\_surface is a Surface at a constant distance from a straight line. The Surface shall always be trimmed to a finite extension either by assigning it to a Face or by specifying parameter values using the objects Rectangular\_trimmed\_surface or Curve\_boundedsurface.

## 4.2.23 Degenerated\_parametric\_curve

A Degenerated\_parametric\_curve is a Parametric\_curve, i.e. a Curve in a 2D parametric space of a reference Surface. In 3D model space, this curve collapses to a single point.

EXAMPLE 3 – This object may be used to model the top of a Conical\_surface.

## 4.2.24 Edge

An Edge is a topological object corresponding to the connection between two Vertexts. It is a finite and non-self-intersecting piece of a curve bounded by Vertexts. An Edge shall always be associated with a geometric Curve.

## 4.2.25 Elementary\_surface

An Elementary\_surface is a collection of analytically defined surface types. Plane, Cylindrical\_surface, Conical\_surface, Spherical\_surface, and Toroidal\_surface are subtypes of Elementary\_surface. The unbounded surface types Plane, Cylindrical\_surface, and Conical\_surface may be combined with Curve\_boundedsurface or Rectangular\_trimmed\_surface, or with topological objects to limit the Surface area and to define voids. Elementary\_surface shall not by itself be instantiated.

## 4.2.26 Ellipse

An Ellipse is a planar closed Curve which may be produced as the intersection of a plane and a conical surface, where the plane is not parallel to any of the straight lines that originate in the top of the cone and that build the conical surface. An Ellipse may be trimmed to an elliptical arc by using a Trimmed\_curve object or the topological objects Vertex and Edge.

## 4.2.27 Face

A Face is a topological object that corresponds to the concept of a piece of a Surface bounded by Loops. A Face may have holes, each hole bounded by a Loop. A Face has a normal N, and the tangent to a Loop is T. For a Loop bounding a Face, the cross product N x T points towards the interior of the Face. A Face shall always be associated with a geometric Surface.

## 4.2.28 Face\_set

A Face\_set is a collection of Faces that are connected with each other.

#### 4.2.29 Geometrically\_boundedsurface\_model

A Geometrically\_boundedsurface\_model is the overall concept of a Surface\_model without explicit topology. It contains a collection of Points, Curves and Surfaces where all relationships among them are geometric of nature.

#### 4.2.30 Geometric\_element

A Geometric\_element is any geometric application object of this part of ISO 10303. These are: Point, Curve, Surface, Axis\_placement, Geometry\_replica, and Transformation. Geometric\_element shall not by itself be instantiated. The dimensionality of a Geometric\_element is 3 with the exception of those Geometric\_elements that are used to define Parametric\_curves.

The data associated with a Geometric\_element are the following:

- user\_defined\_name.

The user\_defined\_name is optional and, if present, specifies the word, or group of words, by which the Geometric\_element is referred to.

#### 4.2.31 Geometry\_replica

A Geometry\_replica describes instances of Curves and Surfaces that are placed in space by applying a Transformation with or without mirroring and uniform scaling. Geometry\_replica may be used recursively, i.e. replicas may again be replicated. The Geometry\_replica is one object even if the original geometry is a collection of objects. Curve\_replica and Surface\_replica are Geometry\_replicas. A Geometry\_replica shall not by itself be instantiated.

#### 4.2.32 Group

A Group is a bag with instances where the elements of the Group have only their membership in common.

The data associated with a Group are the following:

- identifier.

The identifier specifies a word, or group of words, by which the Group is referred to. The identifier shall be unique for all Groups within the scope of an instance population.

#### 4.2.33 Hyperbola

A Hyperbola is a planar Unbounded\_curve which may be produced as the intersection of a plane and a conical surface, where the plane is parallel to two of the straight lines that originate in the top of the cone and that build the conical surface. A Hyperbola shall always be trimmed to a finite length either by parameter values, using a Trimmed\_curve, or by Vertices.

#### **4.2.34 Intersection\_curve**

An **Intersection\_curve** is defined by intersecting two **Surfaces**. It is represented as a **Curve\_3D** with references to two intersecting **Surfaces**.

#### **4.2.35 Layer**

A **Layer** is a grouping mechanism for the visualisation of one or more **Surface\_models**, topological objects and geometric objects. A **Layer** may comprise one or many instances. A **Layer** may contain other **Layers** by including their objects.

The data associated with a **Layer** are the following:

- **user\_defined\_name**.

The **user\_defined\_name** is optional and, if present, specifies the word, or group of words, by which the **Layer** is referred to.

#### **4.2.36 Line**

A **Line** is an unbounded straight curve, i.e. with constant tangent direction. A **Line** shall always be trimmed to a finite length either by parameter values, using the mechanism of **Trimmed\_curve**, or by **Vertices**.

#### **4.2.37 Loop**

A **Loop** is a topological object that consists of either one **Vertex** or a collection of **Edges**. It is indicated whether the orientation of the **Loop** and the **Edge** coincide or not.

EXAMPLE 4 – A **Loop** of **Vertices** has zero extent, and might be used when a **Topology\_surface\_model** includes **Surfaces** with degenerated **Edges**, e.g. triangular **B\_spline\_surfaces**.

#### **4.2.38 Manifold\_surface\_model**

A **Manifold\_surface\_model** is the overall concept of a **Surface\_model** with explicit and manifold topology. **Manifold\_surface\_models** have the following characteristics: they consist of sets of **Shells** where its **Shells** may be open or closed; and they shall not intersect except at **Edges** and **Vertices**. **Shells** may share **Faces**. Distinct **Faces** shall not intersect. Coincident portions of **Shells** shall reference the same **Faces**, **Edges** and **Vertices** that define the coincident region.

#### **4.2.39 Non\_manifold\_surface\_model**

A **Non\_manifold\_surface\_model** is the overall concept of a **Surface\_model** with explicit and non-manifold topology. A **Non\_manifold\_surface\_model** consists of **Face\_sets**. The **Faces** of these sets shall not intersect except at **Edges** and **Vertices**. A **Face** in one set may overlap with a **Face** in another set, provided that the **Face** boundaries are identical.

#### 4.2.40 Offset\_curve

An Offset\_curve is a Curve offset in 3D space. The Offset\_curve may be at any point on the basis Curve in the direction  $V \times T$ , where  $V$  is the fixed reference direction and  $T$  is the unit tangent to the basis Curve. Basis Curve may be any type of Curve. An Offset\_curve shall not self-intersect.

#### 4.2.41 Offset\_surface

An Offset\_surface is a Surface that is defined by offsetting a basis Surface by a distance along the Surface normal of the basis Surface. The offset distance shall not exceed the radius of the curvature in any direction at any point of the basis Surface. An Offset\_surface shall not self-intersect.

#### 4.2.42 Open\_Shell

An Open\_shell is a collection of one or more Faces. Distinct Faces of the Shell do not intersect, but may share Edges. An Open\_shell need not bound a 3D region and, thus, it need not divide a 3D coordinate space into two connected regions. The topological normal of an Open\_shell is the one that would be obtained by filling in the gaps in its domain to produce a Closed\_shell. The normal points from the finite to the infinite. The topological normal may also be derived from the face normals after taking account of their orientation.

#### 4.2.43 Parabola

A Parabola is a planar Unbounded\_curve which may be produced as the intersection of a plane and a conical surface, where the plane is parallel to one of the straight lines that originate in the top of the cone and that build the conical surface. A Parabola shall always be trimmed to a finite length either by parameter values, using the mechanism of Trimmed\_curve, or by Vertices.

#### 4.2.44 Parametric\_curve

A Parametric\_curve is a Curve in the 2D parametric space of a reference Surface. A Parametric\_curve is defined by one Curve\_2d in combination with one Surface.

#### 4.2.45 Part

A Part is a collection of Surface\_models. This collection shall represent a meaningful concept in the context of mechanical design. The collection itself may represent the complete shape of a Product. A Part may be used as one of the leaves of an Assembly and may in this role also be given a location.

The data associated with a Part are the following:

- user\_defined\_name.

The user\_defined\_name specifies the word, or group of words, by which the Part is referred to.

#### 4.2.46 Plane

A Plane is an unbounded Surface with a constant normal. The Surface shall always be trimmed to a finite extension either by parameter values, using the mechanisms of Rectangular\_trimmed\_surface or Curve\_bounded\_surface, or by Edges.

#### 4.2.47 Point

A Point is a location in some coordinate space. Cartesian\_point, Point\_on\_curve, Point\_on\_surface, and Degenerated\_parametric\_curve are subtypes of Point. Point is used in the definitions of other geometric objects such as Curves and Surfaces. A Point shall not by itself be instantiated.

#### 4.2.48 Point\_appearance

A Point\_appearance specifies the visual appearance of a Point.

The data associated with a Point\_appearance are the following:

- colour;
- marker;
- marker\_size.

##### 4.2.48.1 colour

The colour is optional and, if present, defines the basic appearance property with respect to the light reflectance by the marker for a Point. Colour shall be specified by intensity values for red, green, and blue.

##### 4.2.48.2 marker

The marker is optional and, if present, describes the graphical representation of a Point.

##### 4.2.48.3 marker\_size

The marker\_size is optional and, if present, defines the size of the marker.

#### 4.2.49 Point\_on\_curve

A Point\_on\_curve is a Point within a Curve found by evaluating the Curve at a specific parameter value. Point\_on\_curve implicitly represents the topological relation between a Point and a Curve.

#### **4.2.50 Point\_on\_surface**

A Point\_on\_surface is a Point within a Surface found by evaluating the Surface at a particular pair of parameter value. The object thus implicitly represents the topological relation between a Point and a Surface.

#### **4.2.51 Polyline**

A Polyline is a Bounded\_curve defined by a list of n Cartesian\_points and consisting of n-1, but at least two linear segments.

#### **4.2.52 Presentation\_appearance**

A Presentation\_appearance is an aspect of visual presentation that may be assigned to Geometric\_elements, Topological\_elements, and Surface\_models as well as to Layers. The attributes used to define Presentation\_appearance vary depending on the object being presented. Presentation\_appearance is either a Point\_appearance, a Curve\_appearance, or a Surface\_appearance. A Presentation\_appearance shall not by itself be instantiated.

#### **4.2.53 Product**

A Product in the context of this part of ISO 10303 is a physical thing the shape of which is described by one or more Surface\_models. A Product composed of several Parts shall be represented by an Assembly.

The data associated with a Product are the following:

- coordinate\_system;
- user\_defined\_name;
- version\_and\_id.

##### **4.2.53.1 coordinate\_system**

The coordinate\_system specifies the distinct coordinate space for the Product, spatially unrelated to other coordinate spaces. Other coordinate spaces may be related to the one of the Product by means of transformations.

##### **4.2.53.2 user\_defined\_name**

The user\_defined\_name specifies the word, or group of words, by which the Product is referred to.

#### 4.2.53.3 **version\_and\_id**

The **version\_and\_id** specifies version number and identifier that make an instance of a Product uniquely identifiable.

#### 4.2.54 **Rectangular\_composite\_surface**

A **Rectangular\_composite\_surface** is a **Bounded\_surface** that is composed of a rectangular array of segments or patches. Each segment shall be bounded and topologically rectangular, and must, therefore, be either a **B\_spline\_surface** or a **Rectangular\_trimmed\_surface**.

#### 4.2.55 **Rectangular\_trimmed\_surface**

A **Rectangular\_trimmed\_surface** is a **Bounded\_surface** that is constrained by four boundaries which have constant parameter lines. The parameter values should be within the parametric range of the referenced Surface; the referenced Surface may be any subtype of Surface.

#### 4.2.56 **Screen\_image**

A **Screen\_image** is a container of pictures and text. Pictures and text may be presented through different viewports within the same **Screen\_image**. There shall be only one **Screen\_image** present, i.e. all displayed product model information shall be displayed within one **Screen\_image**.

EXAMPLE 5 – A window on a computer display is an example of a **Screen\_image**.

#### 4.2.57 **Seam\_curve**

A **Seam\_curve** is the seam of a Surface that is closed in one parameter direction. The **Seam\_curve** is given by the two **Parametric\_curves** that are coincident at the seam and one **Curve\_3D** that represents the seam in 3D space.

#### 4.2.58 **Shape\_representation**

A **Shape\_representation** is a self-contained and complete description of a shape model. In the context of this part of ISO 10303, the shape model is restricted to be a **Surface\_model**. A **Shape\_representation** shall not by itself be instantiated.

The data associated with a **Shape\_representation** are the following:

- **global\_unit**;
- **shape\_functionality**.

##### 4.2.58.1 **global\_unit**

The **global\_unit** specifies the units that shall be used for all measures of the same kind within a **Shape\_representation**.

EXAMPLE 6 – Unit millimetre may be assigned to all length measures that occur within a Surface\_model.

#### **4.2.58.2 shape\_functionality**

The shape\_functionality is optional and, if present, specifies the functionality of a shape in the context of a Part.

#### **4.2.59 Shell**

A Shell is a Face\_set with additional topological restrictions. A Shell shall be instantiated as either an Open\_shell or a Closed\_shell. A Shell is constructed by joining Faces along Edges, the bounds being included in the domain of the Closed\_shell only. The domain of a Shell is thus a connected 2-manifold, that is made of a connected, oriented, finite, non-self-intersecting Surface that may be open or closed.

#### **4.2.60 Spherical\_surface**

A Spherical\_surface is a Surface at a constant distance from a straight Line.

#### **4.2.61 Surface**

A Surface is a set of connected points in 3D space. Elementary\_surface, Bounded\_surface, Swept\_surface, Offset\_surface, and Surface\_replica are subtypes of Surface. Surface is used in the definition of other geometric objects such as Points and Curves. A Surface shall not be itself instantiated.

#### **4.2.62 Surface\_appearance**

A Surface\_appearance specifies the visual appearance of a Surface.

The data associated with a Surface\_appearance are the following:

- colour;
- grid\_indicator;
- shading\_method.

##### **4.2.62.1 colour**

The colour is optional and, if present, defines the basic appearance property with respect to the light reflectance by the Surface. Colour shall be specified by intensity values for red, green, and blue.

#### 4.2.62.2 grid\_indicator

The `grid_indicator` is optional and, if present, describes the way that a Surface shall be presented. A Surface shall be presented either by shading or by presenting Curves that are either used for the definition of the Surface or are on the Surface. The following self-explaining grid options shall be supported:

- boundary curves;
- silhouette curves;
- segmentation curves;
- control grid;
- parameter line;
- shading.

#### 4.2.62.3 shading\_method

The `Shading_method` is optional and, if present, defines the algorithm that shall be used for shading. Default `Shading_method` is normal shading, i.e. shading based on values from the linear interpolation of the Surface normals.

### 4.2.63 Surface\_model

A `Surface_model` contains the entire or parts of the Surface geometry of a Product. A `Surface_model` is either a `Topology_surface_model`, a `Geometrically_boundedsurface_model`, or a `Surface_model_replica`. A `Surface_model` is a `Shape_representation`. `Surface_model` shall not be instantiated.

The data associated with a `Surface_model` are the following:

- `user_defined_name`.

The `user_defined_name` is optional and, if present, specifies the word, or group of words, by which the `Surface_model` is referred to.

### 4.2.64 Surface\_model\_replica

A `Surface_model_replica` describes an instance of a `Surface_model` that is placed in space by applying the Transformation object with or without mirroring and scaling. `Surface_model_replica` may be used recursively, i.e. replicas may again be replicated. The `Surface_model_replica` is one object even if the original `Surface_model` is a collection of objects.

#### **4.2.65 Surface\_of\_linear\_extrusion**

A Surface\_of\_linear\_extrusion is a Swept\_surface that is defined by sweeping any subtype of Curve, except Parametric\_curve, in a given direction. The result is an infinite Surface. This Surface shall be combined with Curve\_bounded\_surface or Rectangular\_trimmed\_surface, or with topological objects to limit the Surface area. Also voids may be defined using these objects.

#### **4.2.66 Surface\_of\_revolution**

A Surface\_of\_revolution is a Swept\_surface that is defined by rotating any subtype of Curve, except Parametric\_curve, in a complete revolution about an axis. This Surface may be combined with Curve\_bounded\_surface or Rectangular\_trimmed\_surface, or with topological objects to limit the Surface area and to define voids.

#### **4.2.67 Surface\_replica**

A Surface\_replica is the replica of an existing Surface at a different location that is itself a Surface. The Transformation between the original and the Surface\_replica may contain scaling. The Surface\_replica is one object even if the original Surface is a collection of objects. The Points, Curves, and Surfaces of the Surface\_replica are not identified and may not be modified as such.

#### **4.2.68 Swept\_surface**

A Swept\_surface is a collection of Surface types constructed by sweeping any subtype of Curve, except Parametric\_curve, along another Curve. The second Curve shall be either a Circle or a straight Line. These Surfaces may be combined with Curve\_bounded\_surface or Rectangular\_trimmed\_surface, or with topological objects to limit the Surface area and to define voids. Surface\_of\_linear\_extrusion and Surface\_of\_revolution are subtypes of Swept\_surface. Swept\_surface shall not by itself be instantiated.

#### **4.2.69 Topological\_element**

A Topological\_element represents the topology, or connectivity, of objects that make up the representation of another object. A Topological\_element may be any of the topological application objects of this part of ISO 10303. Vertex, Edge, Loop, Face, and Shell are subtypes of Topological\_element. Topological\_element shall not by itself be instantiated.

The data associated with a Topological\_element are the following:

- user\_defined\_name.

The user\_defined\_name is optional and, if present, specifies the word, or group of words, by which the Topological\_element is referred to.

#### **4.2.70 Topology\_surface\_model**

A Topology\_surface\_model represents a Surface\_model that uses topological constructs such as Vertex, Edge, and Face. A Topology\_surface\_model is a Non\_manifold\_surface\_model or a Manifold\_surface\_model. A Topology\_surface\_model shall not by itself be instantiated.

#### **4.2.71 Toroidal\_surface**

A Toroidal\_surface is a Surface which can be produced by revolving a Circle about a non-intersecting Line in its plane.

#### **4.2.72 Transformation**

A Transformation is a geometric transformation composed of translation, rotation, mirroring and uniform scaling. Transformation is used for replicating 3D geometric objects and for specifying Assemblies.

#### **4.2.73 Trimmed\_curve**

A Trimmed\_curve is a Bounded\_curve that is a Curve defined by taking the portion between two Points on the associated basis Curve. The trimming Points are defined as parameter values in the domain of the basis Curve. The basis Curve may be any type of Curve.

#### **4.2.74 Unbounded\_curve**

An Unbounded\_curve is a collection of curve types with no natural start or end Points. Line, Circle, Ellipse, Hyperbola and Parabola are subtypes of Unbounded\_curve. Unbounded\_curves shall be trimmed to get a finite length. An Unbounded\_curve shall not by itself be instantiated.

#### **4.2.75 Vertex**

A Vertex is the topological construct that corresponds to a Point. It must always reference a geometric Point and shall not by itself be instantiated. A Vertex may be used to trim an Edge. A Vertex may also define a Loop that only consists of one Vertex.

### **4.3 Application assertions**

This subclause specifies the application assertions for the mechanical design using surface representation application protocol. Application assertions specify the relationships between application objects, the cardinality of the relationships, and the rules required for the integrity and validity of the application objects and UoFs. The application assertions and their definitions are given below.

#### **4.3.1 3D\_projection to Screen\_image**

A 3D\_projection is in one Screen\_image. A Screen\_image contains one or many 3D\_projections.

#### **4.3.2 3D\_projection to Surface\_model**

A 3D\_projection represents one Surface\_model. A Surface\_model exists as zero, one, or many 3D\_projections.

#### **4.3.3 Annotation\_text to Screen\_image**

An Annotation\_text is presented in one Screen\_image. A Screen\_image contains zero, one, or many Annotation\_texts.

#### **4.3.4 Annotation\_text to Transformation**

An Annotation\_text is located by one and exactly one transformation. Each Transformation locates zero, one or many Annotation\_texts.

#### **4.3.5 Assembly to Assembly**

An Assembly consists of zero, one, or many Assemblies. An Assembly is part of zero, one, or many Assemblies.

#### **4.3.6 Assembly to Layer**

An Assembly is assigned to zero, one, or many Layers. A Layer contains zero, one, or many Assemblies.

#### **4.3.7 Assembly to Part**

An Assembly consists of one or many Parts. Each Part is included in zero, one, or many Assemblies.

#### **4.3.8 Assembly to Product**

An Assembly contains zero, one, or many Products. A Product is part of zero, one, or many Assemblies. An Assembly is part of one or many Products. A Product consists of zero, one, or many Assemblies.

#### **4.3.9 Assembly to Transformation**

An Assembly is located by zero or one Transformation. A Transformation locates zero, one, or many Assemblies.

### **4.3.10 Composite\_curve\_on\_surface to Intersection\_curve**

A Composite\_curve\_on\_surface is defined by one or many Intersection\_curves. An Intersection\_curve defines zero, one, or many Composite\_curve\_on\_surfaces.

### **4.3.11 Composite\_curve\_on\_surface to Seam\_curve**

A Composite\_curve\_on\_surface is defined by one or many Seam\_curves. A Seam\_curve defines zero, one, or many Composite\_curve\_on\_surfaces.

### **4.3.12 Composite\_curve\_on\_surface to Parametric\_curve**

A Composite\_curve\_on\_surface is defined by one or many Parametric\_curves. A Parametric\_curve defines zero, one, or many Composite\_curve\_on\_surfaces.

### **4.3.13 Curve to Curve\_appearance**

A Curve is presented by zero or one Curve\_appearance. A Curve\_appearance presents zero, one, or many Curves.

### **4.3.14 Curve to Curve\_replica**

A Curve is master of zero, one, or many Curve\_replicas. A Curve\_replica is instance of one and exactly one Curve.

### **4.3.15 Curve to Edge**

A Curve is in zero, one, or many Edges. An Edge has one and exactly one Curve .

### **4.3.16 Curve to Geometrically\_boundedsurface\_model**

A Curve is part of zero, one, or many Geometrically\_boundedsurface\_models. A Geometrically\_boundedsurface\_model consists of zero, one, or many Curves.

### **4.3.17 Curve\_2D to Parametric\_curve.**

A Curve\_2D partly defines one or many Parametric\_curves. A Parametric\_curve is defined by one and exactly one Curve\_2D in combination with one and exactly one Surface.

### **4.3.18 Curve\_3D to Intersection\_curve**

A Curve\_3D partly defines zero, one, or many Intersection\_curves. An Intersection\_curve is defined by one and exactly one Curve\_3D and exactly two Surfaces.

#### **4.3.19 Curve\_3D to Seam\_curve**

A Curve\_3D partly defines zero, one, or many Seam\_curves. A Seam\_curve is defined by one and exactly one Curve\_3D in combination with exactly two Parametric\_curves.

#### **4.3.20 Curve\_appearance to Edge**

A Curve\_appearance presents zero, one, or many Edges. An Edge is presented by zero or one Curve\_appearance.

#### **4.3.21 Edge to Loop**

An Edge belongs to one or two Loops. A Loop has zero, one or many Edges.

#### **4.3.22 Edge to Vertex**

An Edge has exactly two references to Vertex. A Vertex belongs to zero, one, or many Edges.

#### **4.3.23 Face to Face\_set**

A Face is part of one or many Face\_sets. A Face\_set consists of one or many Faces.

#### **4.3.24 Face to Loop**

A Face has one or many Loops. A Loop belongs to one and exactly one Face.

#### **4.3.25 Face to Surface**

A Face has one and exactly one Surface. A Surface belongs to zero, one, or many Faces.

#### **4.3.26 Face to Surface\_appearance**

A Face is presented by zero or one Surface\_appearance. A Surface\_appearance presents zero, one, or many Faces.

#### **4.3.27 Face\_set to Manifold\_surface\_model**

A Face\_set is part of zero, one, or many Manifold\_surface\_models. A Manifold\_surface\_model consists of one or many Face\_sets.

### **4.3.28 Face\_set to Non\_manifold\_surface\_model**

A Face\_set is part of zero, one, or many Non\_manifold\_surface\_models. A Non\_manifold\_surface\_model consists of one or many Face\_sets.

### **4.3.29 Face\_set to Presentation\_appearance**

A Face\_set is presented by zero, one, or many Presentation\_appearances. A Presentation\_appearance presents zero, one, or many Face\_sets.

### **4.3.30 Geometrically\_boundedsurface\_model to Point**

A Geometrically\_boundedsurface\_model consists of zero, one, or many Points. A Point is part of zero, one, or many Geometrically\_boundedsurface\_models.

### **4.3.31 Geometrically\_boundedsurface\_model to Surface**

A Geometrically\_boundedsurface\_model consists of zero, one, or many Surfaces. A Surface is part of zero, one, or many Geometrically\_boundedsurface\_models.

### **4.3.32 Geometric\_element to Group**

A Geometric\_element belongs to zero, one, or many Groups. A Group contains zero, one, or many Geometric\_elements.

### **4.3.33 Geometric\_element to Layer**

A Geometric\_element is assigned to zero, one, or many Layers. A Layer contains zero, one, or many Geometric\_elements.

### **4.3.34 Geometry\_replica to Transformation**

A Geometry\_replica is located by one and exactly one Transformation. A Transformation locates zero, one, or many Geometry\_replicas.

### **4.3.35 Group to Surface\_model**

A Group contains zero, one, or many Surface\_models. A Surface\_model belongs to zero, one, or many Groups.

### **4.3.36 Group to Topological\_element**

A Group contains zero, one, or many Topological\_elements. A Topological\_element belongs to zero, one, or many Groups.

#### **4.3.37 Intersection\_curve to Surface**

An Intersection\_curve is defined by two and exactly two Surfaces in combination with exactly one Curve\_3D. A Surface partly defines zero, one, or many Intersection\_curves.

#### **4.3.38 Layer to Layer**

A Layer contains zero, one, or many Layers. A Layer is assigned to zero, one, or many Layers.

#### **4.3.39 Layer to Part**

A Layer contains zero, one, or many Parts. A Part is assigned to zero, one, or many Layers.

#### **4.3.40 Layer to Presentation\_appearance**

A Layer is presented by zero, one, or many Presentation\_appearances. A Presentation\_appearance presents zero, one, or many Layers.

#### **4.3.41 Layer to Product**

A Layer contains zero, one, or many Products. A Product is assigned to zero, one, or many Layers.

#### **4.3.42 Layer to Surface\_model**

A Layer contains zero, one, or many Surface\_models. A Surface\_model is assigned to zero, one, or many Layers.

#### **4.3.43 Layer to Topological\_element**

A Layer contains zero, one, or many Topological\_elements. A Topological\_element is assigned to zero, one, or many Layers.

#### **4.3.44 Loop to Vertex**

A Loop consists of zero or one Vertex. A Vertex belongs to zero or one Loop.

#### **4.3.45 Parametric\_curve to Seam\_curve**

A Parametric\_curve partly defines zero, one, or many Seam\_curves. A Seam\_curve is defined by two and exactly two Parametric\_curves in combination with one and exactly one Curve\_3D.

#### **4.3.46 Parametric\_curve to Surface**

A Parametric\_curve is defined by exactly one Surface that is the basis for exactly one Curve\_2D. A Surface contains zero, one, or many Parametric\_curves.

#### **4.3.47 Part to Product**

A Part is part of zero, one, or many Products. A Product consists of zero, one, or many Parts.

#### **4.3.48 Part to Shape\_representation**

A Part has one or many Shape\_representations. A Shape\_representation is shape of one or many Parts.

#### **4.3.49 Part to Transformation**

A Part is located by zero or one Transformation. A Transformation locates zero, one, or many Parts.

#### **4.3.50 Point to Point\_appearance**

A Point is presented by zero or one Point\_appearance. A Point\_appearance presents zero, one, or many Points.

#### **4.3.51 Point to Vertex**

A Point is in zero, one, or many instances of type Vertex. A Vertex has exactly one Point.

#### **4.3.52 Presentation\_appearance to Surface\_model**

A Presentation\_appearance presents zero, one, or many Surface\_models. A Surface\_model is presented by zero, one, or many Presentation\_appearances.

#### **4.3.53 Surface to Surface\_appearance**

A Surface is presented by zero or one Surface\_appearance. A Surface\_appearance presents zero, one, or many Surfaces.

#### **4.3.54 Surface to Surface\_replica**

A Surface is master of zero, one, or many Surface\_replicas. A Surface\_replica is instance of exactly one Surface.

#### **4.3.55 Surface\_model to Surface\_model\_replica**

A Surface\_model is copied to zero, one, or many Surface\_model\_replicas. A Surface\_model\_replica is instance of exactly one Surface\_model.

#### **4.3.56 Surface\_model\_replica to Transformation**

A Surface\_model\_replica is located by one and exactly one Transformation. A Transformation locates zero, one, or many Surface\_model\_replicas.

## 5 Application interpreted model

### 5.1 Mapping table

This clause contains the mapping table that shows how each UoF and application object of this part of ISO 10303 (see clause 4) maps to one or several resource constructs (see 5.2).

The mapping table is organized in five columns. The contents of these five columns are:

- Column 1) Application element: Name of an application element as it appears in the application object definition in 4.2. Application object names are written in uppercase. Attribute names and assertions are listed after the application object to which they belong and are written in lower case.
- Column 2) AIM element: Name of an AIM element as it appears in the AIM (see 5.2). AIM entities are written in lower case. Attribute names of AIM entities are referred to as <entity name> . <attribute name>. The mapping of an application element may result in several related AIM elements. Each of these AIM elements requires a line of its own in the table.
- Column 3) Source: For those AIM elements that are interpreted from the integrated resources, this is the number of the corresponding part of ISO 10303. For those AIM elements that are created for the purpose of this part of ISO 10303, this is the number of the part being written. For those AIM elements that are directly incorporated from an application interpreted construct (AIC) this is the AIC reference.
- Column 4) Rules: One or more numbers may be given that refer to rules that apply to the current AIM element. For rules that are derived from relationships between application objects, the same rule is referred to by the mapping entries of all the involved AIM elements. Global rules are given for each application object or AIM element to which they apply. Immediately following the table, the rules are listed by number and by expanded name.
- Column 5) Reference path: To describe fully the mapping of an ARM element it may be necessary to specify a reference path through several related AIM elements. A single AIM element is documented on a single row within the reference path column with a symbol that defines its relationship to the AIM element on the succeeding row in the column. The reference path column, therefore, documents the role of an AIM element relative to the AIM element in the row succeeding it. Two or more such related AIM elements define the interpretation of the integrated resources that satisfies the requirement specified by the ARM element if a reference path is provided.

For each AIM element that has been created for use within this part of ISO 10303, a reference path up to its supertype from an integrated resource or an AIC is specified. In the case of a bi-directional reference from an AIM element with two attributes, each of which spawns a reference path, each reference path is enclosed by a set of parentheses. The AIM element that is the root of both reference paths is documented either between the sets of parentheses or preceding each set of parentheses.

## **ISO/CD 10303-205**

For the expression of reference paths the following notational conventions apply:

- a)  $->$  : attribute references the entity or select type in the following row;
- b)  $<-$  : attribute is entity or select type referenced by the attribute in the following row;
- c)  $=>$  : entity is a supertype of the entity in the following row;
- d)  $<=$  : entity is a subtype of the entity in the following row;
- e)  $=$  : attribute has as its type the specified select or enumeration type, possibly constrained to particular choices or values.

This part of ISO 10303 makes use of the following application interpreted constructs (AIC):

- AIC1 = AIC for Geometrically Bounded Surface Representations;
- AIC2 = AIC for Manifold Surface Representations;
- AIC3 = AIC for Non-manifold Surface Representations;
- AIC4 = AIC for Mechanical Design Context;
- AIC5 = AIC for Mechanical Design Presentation.

Table 2 – Mapping table for geometrically\_boundedsurface\_modelshape UoF

Application element	AIM element	Source	Rules	Reference path
geometrically_boundedsurface_modelshape UoF				
AXIS_PLACEMENT	placement	AIC1		
B_SPLINE_CURVE	b_spline_curve_with_knots OR uniform_curve OR quasi_uniform_curve OR bezier_curve OR rational_b_spline_curve	AIC1 AIC1 AIC1 AIC1 AIC1		
B_SPLINE_SURFACE	b_spline_surface_with_knots OR uniform_surface OR quasi_uniform_surface OR bezier_surface OR rational_b_spline_surface	AIC1 AIC1 AIC1 AIC1 AIC1		
BOUNDED_CURVE	bounded_curve	AIC1		
BOUNDED_SURFACE	bounded_surface	AIC1		
CARTESIAN_POINT	cartesian_point	AIC1		
CIRCLE	circle	AIC1		
COMPOSITE_CURVE	composite_curve	AIC1		

Table 2 (continued)

Application element	AIM element	Source	Rules	Reference path
geometrically_boundedsurface_model_shape_UoF				
COMPOSITE_CURVE_ON_SURFACE	composite_curve_on_surface	AIC1		composite_curve_on_surface ⇌ composite_curve
composite_curve_on_surface to intersection_curve				composite_curve.segments[i] → composite_curve_segment {reparametrised_composite_ curve_segment}
				composite_curve_segment. parent_curve
				curve surface_curve intersection_curve
composite_curve_on_surface to seam_curve				composite_curve_on_surface ⇌ composite_curve composite_curve.segments[i] → composite_curve_segment composite_curve_segment. parent_curve
				curve surface_curve seam_curve
composite_curve_on_surface to parametric_curve				composite_curve_on_surface ⇌ composite_curve composite_curve.segments[i] → composite_curve_segment

Table 2 (continued)

Application element	AIM element	Source	Rules	Reference path
<b>geometrically_boundedsurface_model_shape_UoF</b>				
				composite_curve_segment.parent_curve curve pcurve
CONICAL_SURFACE	conical_surface	AIC1		↑ curve pcurve
CURVE	curve	AIC1		↓ curve curve.replica.parent_curve curve.replica
curve to curve_replica				
curve to geometrically_boundedsurface_model				= curve geometric_set.select geometric_set.elements[i] geometric_set
CURVE_2D	curve	AIC1		↓ curve geometric_representation_item representation_item definitional_representation_item.items[j] definitional_representation_item parametric_representation_context pcurve.reference_to_curve pcurve
curve_2d to parametric_curve				
CURVE_3D	curve	AIC1		curve surface_curve.curve_3d
curve_3d to intersection_curve				↓

Table 2 (continued)

Application element	AIM element	Source	Rules	Reference path
geometrically_boundedsurface_modelshape_UoF				
curve_3d to seam_curve			surface_curve intersection_curve ⇒	
CURVE_BOUNDED_SURFACE	curve_boundedsurface	AIC1		
CURVE_ON_SURFACE	curve_on_surface	AIC1		
CURVE_REPLICA	curve_replica	AIC1		
CYLINDRICAL_SURFACE	cylindrical_surface	AIC1		
DEGENERATED_PARAMETRIC_CURVE	degenerate_pc当地 OR evaluated_degenerate_pc当地	AIC1 AIC1		
ELEMENTARY_SURFACE	elementary_surface	AIC1		
ELLIPSE	ellipse	AIC1		
GEOMETRICALLY_BOUNDED_SURFACE_MODEL	geometric_set	AIC1	rule!	
geometrically_boundedsurface_model to point			geometric_set,elements[i] → geometric_set_select = point	
geometrically_boundedsurface_model to surface			geometric_set geometric_set,elements[i] → geometric_set_select = surface	

Table 2 (continued)

Application element	AIM element	Source	Rules	Reference path
<b>geometrically_boundedsurface_modelshape_UoF</b>				
GEOMETRIC_ELEMENT user_defined_name	geometric_representation_item mechanical_design_name_assignment.assigned_name	43 205	2,15,19	geometric_representation_item named_item_select mechanical_design_name_assignment. items[i] mechanical_design_name_assignment name_assignment name_assignment.assigned_name
GEOOMETRY_REPLICA	curve_replica OR surface_replica	AIC1 AIC1		(curve_replica curve_replica.transformation) (surface_replica surface_replica.transformation) cartesian_transformation_operator cartesian_transformation_operator_3d
geometry_replica_to_transformation				↑ ⇒
HYPERBOLA	hyperbola	AIC1		
INTERSECTION_CURVE intersection_curve_to_surface	intersection_curve	AIC1		intersection_curve surface_curve surface_curve.associated_geometry[i] → pcurve_or_surface (surface) (pcurve pcurve,basis_surface surface)

Table 2 (continued)

Application element	AIM element	Source	Rules	Reference path
geometrically_boundedsurface_model_shape_UoF				
LINE	line	AIC1		
OFFSET_CURVE	offset_curve_3d	AIC1		
OFFSET_SURFACE	offset_surface	AIC1		
PARABOLA	parabola	AIC1		
PARAMETRIC_CURVE	pcurve	AIC1		
parametric_curve to seam_curve			pcurve pcurve_on_surface surface_curve_associated_geometry[i] surface_curve seam_curve	= ↓ ⇒
parametric_curve to surface			pcurve pcurve_basis_surface	↑
PLANE	plane	AIC1		
POINT	point	AIC1		
POINT_ON_CURVE	point_on_curve	AIC1		
POINT_ON_SURFACE	point_on_surface	AIC1		
POLYLINE	polyline	AIC1		
RECTANGULAR_COMPOSITE_SURFACE	rectangular_composite_surface	AIC1		
RECTANGULAR_TRIMMED_SURFACE	rectangular_trimmed_surface	AIC1		
SEAM_CURVE	seam_curve	AIC1		
SPHERICAL_SURFACE	spherical_surface	AIC1		

Table 2 (continued)

Application element	AIM element	Source	Rules	Reference path
geometrically_boundedsurface_modelshape_UoF				
SURFACE	surface	AIC1		
SURFACE_MODEL	geometrically_boundedsurface_shape_representation	AIC1		
user_defined_name	mechanical_design_name_assignment.assigned_name	205	geometrically_boundedsurface_shape_representation shape_representation named_item_select mechanical_design_name_assignment. items[i] mechanical_design_name_assignment name_assignment name_assignment.assigned_name	↔ =   ↓ ↔ ↔ ↔
SURFACE_MODEL_REPLICA	geometric_set_replica	AIC1	mapped_item mapped_item mapped_item.mapping_source representation_map representation_map. mapped_representation representation shape_representation geometrically_boundedsurface_shape_representation	→ → → → → ↔ ↔
surface_model_replica_to_surface_model			1,9	

Table 2 (concluded)

Application element	AIM element	Source	Rules	bf Reference path
<b>geometrically_boundedsurface_modelshape_UoF</b>				
surface.model.replica_to_transformation				geometric.set.replica geometric.set.replica geometric.set.replica.transformation → cartesian_transformation.operator ⇒ cartesian_transformation.operator_3d
SURFACE_OF_LINEAR_EXTRUSION	surface_of_linear_extrusion	AIC1		
SURFACE_OF_REVOLUTION	surface_of_revolution	AIC1		
SURFACE_REPLICA	surface.replica	AIC1		
surface.replica_to_surface				surface.replica surface.replica.parent_surface → surface
SWEPT_SURFACE	swept_surface	AIC1		
TOROIDAL_SURFACE	toroidal_surface	AIC1		
TRANSFORMATION	cartesian_transformation_operator_3d	AIC1		
TRIMMED_CURVE	trimmed_curve	AIC1		
UNBOUNDED_CURVE	line	AIC1		

Table 3 – Mapping table for grouping UoF

Application element	AIM element	Source	Rules	Reference path
grouping UoF				
GROUP	group	41		
identifier	group.name	41		
group to geometric_element				group group_assignment.assigned_group group_assignment mechanical_design_group_assignment mechanical_design_group_assignment.items[i] group_item_select shape_representation geometric_representation.item
group to surface_model			1,9	group group_assignment.assigned_group group_assignment mechanical_design_group_assignment mechanical_design_group_assignment.items[i] group_item_select shape_representation (geometrically_bound_surface_shape_representation) (manifold_surface_shape_representation) (non_manifold_surface_shape_representation)
group to topological_element			1,9 11	group group_assignment.assigned_group group_assignment mechanical_design_group_assignment mechanical_design_group_assignment.items[i] group_item_select shape_representation topological_representation.item

Table 4 – Mapping table for manifold\_surface\_model\_shape\_UoF

Application element	AIM element	Source	Rules	Reference path
manifold_surface_model_shape_UoF				
AXIS_PLACEMENT	placement	AIC2		
B_SPLINE_CURVE	b_spline_curve_with_knots OR uniform_curve OR quasi_uniform_curve OR bezier_curve OR rational_b_spline_curve	AIC2 AIC2 AIC2 AIC2 AIC2		
B_SPLINE_SURFACE	b_spline_surface-with_knots OR uniform_surface OR quasi_uniform_surface OR bezier_surface OR rational_b_spline_surface	AIC2 AIC2 AIC2 AIC2 AIC2		
BOUNDED_CURVE	bounded_curve	AIC2		
BOUNDED_SURFACE	bounded_surface	AIC2		
CARTESIAN_POINT	cartesian_point	AIC2		
CIRCLE	circle	AIC2		
CLOSED_SHELL	closed_shell	AIC2		
CONICAL_SURFACE	conical_surface	AIC2		

Table 4 (continued)

Application element	AIM element	Source	Rules	Reference path
	manifold_surface.modelshape_UoF			
CURVE		AIC2		
curve to curve_replica	curve		curve curve_replica.parent_curve curve_replica	↓
curve to edge			curve edge_curve.edge_geometry edge_curve edge	↓ ↔
CURVE_2D	curve	AIC2		
curve_2d to parametric_curve			curve geometric_representation_item representation_item definitional_representation_item.items[i] definitional_representation_item parametric_representation_context pcurve.reference_to_curve pcurve	↓ ↔ ↓ ↓ ↑ ↓
CURVE_3D	curve	AIC2		
curve_3d to intersection_curve			curve surface_curve.curve_3d surface_curve intersection_curve	↓ ↑
curve_3d to seam_curve			surface_curve.curve_3d surface_curve seam_curve	↓ ↑

Table 4 (continued)

Application element	AIM element	Source	Rules	Reference path
manifold_surface.model_shape_UoF				
CURVE_ON_SURFACE	surface_curve OR pcurve	AIC2		
CURVE_REPLICA	curve_replica	AIC2		
CYLINDRICAL_SURFACE	cylindrical_surface	AIC2		
DEGENERATED_PARAMETRIC_CURVE	degenerate_pcurve OR evaluated_degenerate_pcurve	AIC2		
EDGE	edge	AIC2	edge (edge.edge_start) (edge.edge_end) → vertex vertex.point	
edge to vertex				
ELEMENTARY_SURFACE	elementary_surface	AIC2		
ELLIPSE	ellipse	AIC2		
FACE	advanced_face OR oriented_face OR face_surface	AIC2 AIC2 AIC2		

Table 4 (continued)

Application element	AIM element	Source	Rules	Reference path
manifold_surface_model_shape_UoF				
face to loop	AIC2		(advanced_face face_surface) (oriented_face) (face_surface) face face_bounds[i] face_bound face_outer_bound face_bound_bound loop (edge_loop) (vertex_loop)	⇒ ⇒ → {⇒ ↑ ↑ ↑ ⇒
face to surface	AIC2		(advanced_face face_surface) (oriented_face) oriented_face, face_element face face_surface (face_surface) face_surface.face_geometry surface	⇒ ↑ ↑ ⇒
FACE_SET	connected_face_set	AIC2		
face_set to face		AIC2	connected_face_set connected_face_set.cfs.faces[1] face (oriented_face) (face_surface) advanced_face)	→ ⇒ ⇒

Table 4 (continued)

Application element	AIM element	Source	Rules	Reference path
manifold_surface.model_shape_UoF				
GEOMETRIC_ELEMENT	geometric_representation_item	43	2,15,19	
user_defined_name	mechanical_design.name_assignment.assigned_name	205	geometric_representation_item named_item.select mechanical_design.name_assignment. items[i]	= ← →
GEOMETRY_REPLICA	curve_replica OR surface_replica	AIC2	(curve_replica curve.replica.transformation) (surface_replica) surface.replica.transformation cartesian_transformation_operator ⇒ cartesian_transformation_operator_3d	
HYPERBOLA	hyperbola	AIC2		
INTERSECTION_CURVE	intersection_curve	AIC2	intersection_curve surface_curve surface_curve.associated_geometry[i] pcurve_or_surface (pcurve pcurve.basis_surface surface)	← → =

Table 4 (continued)

Application element	AIM element	Source	Rules	Reference path
manifold_surface_model_shape_UoF				
LINE	line	AIC2		
LOOP	loop	AIC2		
	vertex_loop OR edge_loop	AIC2		
loop to edge			edge_loop path path.edge_list[i] oriented_edge edge	↔ ↑ ↔
loop to vertex			vertex_loop vertex_loop.loop_vertex vertex vertex_point	↑ ⇒
MANIFOLD_SURFACE_MODEL	shell_based_surface_model	AIC2		
manifold_surface_model to face_set			shell_based_surface_model shell_based_surface_model. sbsm_boundary[i] shell (open_shell connected_face_set) (closed_shell connected_face_set)	↑ = ↔ ↓
OFFSET_CURVE	offset_curve_3d	AIC2		

Table 4 (continued)

Application element	AIM element	Source	Rules	Reference path
manifold_surface_model_shape_U_OF				
OFFSET_SURFACE	offset_surface	AIC'2		
OPEN_SHELL	open_shell	AIC'2		
PARABOLA	parabola	AIC'2		
PARAMETRIC_CURVE	pcurve	AIC'2		
parametric_curve to seam_curve			pcurve pcurve_or_surface surface_curve.associated_geometry[i] surface_curve seam_curve	= ↳ ⇒
parametric_curve to surface			pcurve pcurve,basis_surface	↑ →
PLANE	plane	AIC'2		
POINT	point	AIC'2		
point to vertex			point vertex_point.vertex_geometry vertex_point	↳
POINT_ON_CURVE	point_on_curve	AIC'2		
POINT_ON_SURFACE	point_on_surface	AIC'2		
SEAM_CURVE	seam_curve	AIC'2		

Table 4 (continued)

Application element	AIM element	Source	Rules	Reference path
manifold_surface_model_shape_UoF				
SHELL shell to face	connected_face_set	AIC2		connected_face_set → face {⇒ (face_surface advanced_face) (oriented_face)}
SPHERICAL_SURFACE	spherical_surface	AIC2		
SURFACE	surface	AIC2		
SURFACE_MODEL user_defined_name	manifold_surface_shape_representation mechanical_design_name_assignment.assigned_name	AIC2 205	1,9 manifold_surface_shape_representation = named_item_select mechanical_design_name_assignment. items[]	↓ = ↓ mechanical_design_name_assignment name_assignment name_assignment.assigned_name
SURFACE_MODEL_REPLICA surface_model_replica to surface_model	mapped_item	43	3 mapped_item mapped_item.mapping_source representation_map representation_map. mapped_representation representation	↑ ↑ ↑ ↑

Table 4 (continued)

Application element	AIM element	Source	Rules	Reference path
manifold_surface_model_shape_UoF				
SURFACE_OF_LINEAR_EXTRUSION	surface_of_linear_extrusion	AIC2	1,9 shape_representation manifold_surface. shape_representation →	
SURFACE_OF_REVOLUTION	surface_of_revolution	AIC2		
SURFACE_REPLICA	surface_replica	AIC2		
surface_replica_to_surface			surface_replica.parent_surface surface	→
SWEPT_SURFACE	swept_surface	AIC2		
TOPOLOGICAL_ELEMENT	topological_representation_item	AIC2	11	
user_defined_name	mechanical_design_name_assignment.assigned_name	205	topological_representation_item = named.item_select items[j] mechanical_design_name_assignment. mechanical_design_name_assignment name_assignment name_assignment.assigned_name ←	↔
TOPOLOGY_SURFACE_MODEL	shell_based_surface_model	AIC2		
TOROIDAL_SURFACE	toroidal_surface	AIC2		

Table 4 (concluded)

Application element	AIM element	Source	Rules	Reference path
	manifold_surface.model_shape_UoF			
TRANSFORMATION	cartesian_transformation_operator_3d OR (mapped_item.mapping_target AND mapped_item.mapping_source.mapping_origin)	AIC2 43 43	3 3 3	
UNBOUNDED_CURVE	line	AIC2		
VERTEX	vertex_point	AIC2		

Table 5 – Mapping table for non\_manifold\_surface\_model\_shape UoF

Application element	AIM element	Source	Rules	Reference path
non_manifold_surface_model_shape UoF				
AXIS_PLACEMENT	placement	AIC3		
B_SPLINE_CURVE	b_spline_curve_with_knots OR uniform_curve OR quasi_uniform_curve OR bezier_curve OR rational_b_spline_curve	AIC3 AIC3 AIC3 AIC3 AIC3		
B_SPLINE_SURFACE	b_spline_surface-with_knots OR uniform_surface OR quasi_uniform_surface OR bezier_surface OR rational_b_spline_surface	AIC3 AIC3 AIC3 AIC3 AIC3		
BOUNDED_CURVE	bounded_curve	AIC3		
BOUNDED_SURFACE	bounded_surface	AIC3		
CARTESIAN_POINT	cartesian_point	AIC3		
CIRCLE	circle	AIC3		
CLOSED_SHELL	closed_shell	AIC3		
CONICAL_SURFACE	conical_surface	AIC3		

Table 5 (continued)

Application element	AIM element	Source	Rules	Reference path
	non_manifold_surface_modelshape_UoF			
CURVE	curve	AIC3		curve curve_replica.parent_curve curve_replica
curve to curve_replica				curve curve_to_curve
curve to edge				curve edge_curve.edge_geometry edge_curve edge
CURVE_2D	curve	AIC3		curve geometric_representation_item representation_item definitional_representation_item.items[i] definitional_representation_item parametric_representation_context pcurve.reference_to_curve pcurve
curve_2d to parametric_curve				curve geometric_representation_item representation_item definitional_representation_item.items[i] definitional_representation_item parametric_representation_context pcurve.reference_to_curve pcurve
CURVE_3D	curve	AIC3		curve surface_curve.curve_3d surface_curve intersection_curve
curve_3d to intersection_curve				curve surface_curve.curve_3d surface_curve seam_curve
curve_3d to seam_curve				curve surface_curve.curve_3d surface_curve seam_curve

Table 5 (continued)

Application element	AIM element	Source	Rules	Reference path
non_manifold_surface_model_shape_UoF				
CURVE_ON_SURFACE	surface_curve OR pcurve	AIC3		
CURVE_REPLICA	curve_replica	AIC3		
CYLINDRICAL_SURFACE	cylindrical_surface	AIC3		
DEGENERATED_PARAMETRIC_CURVE	degenerate_pcurve OR evaluated_degenerate_pcurve	AIC3		
EDGE	edge	AIC3	edge (edge.edge_start) (edge.edge_end) → vertex vertex.point	
edge to vertex				
ELEMENTARY_SURFACE	elementary_surface	AIC3		
ELLIPSE	ellipse	AIC3		
FACE	advanced_face OR oriented_face OR face_surface	AIC3 AIC3 AIC3		

Table 5 (continued)

Application element	AIM element	Source	Rules	Reference path
non_manifold_surface_modelshape_UoF				
face to loop	AIC3	(advanced_face face_surface) (oriented_face) (face_surface) face face_bounds[i] face_bound face_outer_bound} face_bound.bound loop (edge_loop) (vertex_loop)	⇓ ⇓ ↑ {⇒ ↑ ↑ ↑ ⇒ ↑ ↑	
face to surface	AIC3	(advanced_face face_surface) (oriented_face) oriented_face.face_element face face_surface) (face_surface) face_surface.face_geometry surface	⇓ ⇒ ↑ ↑ ↑	
FACE_SET	connected_face_set	AIC3		
face_set to face		AIC3	connected_face_set connected_face_set.clf_faces[1] face (oriented_face) (face_surface advanced_face)	↑ ↑ ↑

Table 5 (continued)

Application element	AIM element	Source	Rules	Reference path
non_manifold_surface_model_shape_UoF				
GEOMETRIC_ELEMENT	geometric_representation_item	43	2,15,19	
user_defined_name	mechanical_design.name_assignment.assigned_name	205	geometric_representation_item named_item.select mechanical_design.name_assignment. items[i]	= ← →
GEOMETRY_REPLICA	curve_replica OR surface_replica	AIC3	(curve_replica curve.replica.transformation) (surface_replica) surface.replica.transformation cartesian_transformation_operator ⇒ cartesian_transformation_operator_3d	
HYPERBOLA	hyperbola	AIC3		
INTERSECTION_CURVE	intersection_curve	AIC3	intersection_curve surface_curve surface_curve.associated_geometry[i] pcurve_or_surface (pcurve pcurve.basis_surface surface)	↔ → = →

Table 5 (continued)

Application element	AIM element	Source	Rules	bf Reference path
non_manifold_surface_model_shape_UoF				
LINE	line	AIC3		
LOOP	loop	AIC3		
	vertex_loop OR edge_loop	AIC3		
loop to edge			edge_loop path path.edge_list[i] oriented_edge edge	↔ → ↔
loop to vertex			vertex_loop vertex.loop.vertex vertex vertex_point	→ → →
NON_MANIFOLD_SURFACE_MODEL	face_based_surface_model	AIC3	face_based_surface_model face_based_surface_model. fbsm_faces[i] connected_face_set	→

Table 5 (continued)

Application element	AIM element	Source	Rules	Reference path
non_manifold_surface_modelshape_UoF				
OFFSET_CURVE	offset_curve_3d	AI/C3		
OFFSET_SURFACE	offset_surface	AI/C3		
OPEN_SHELL	open_shell	AI/C3		
PARABOLA	parabola	AI/C3		
PARAMETRIC_CURVE	pcurve	AI/C3		
parametric_curve to seam_curve			pcurve pcurve_or_surface surface_curve_associated_geometry[i] surface_curve seam_curve	= ↳ ⇒
parametric_curve to surface			pcurve pcurve_basis_surface surface	→
PLANE	plane	AI/C3		
POINT	point	AI/C3		
point to vertex			point vertex_point.vertex_geometry vertex_point	↓
POINT_ON_CURVE	point_on_curve	AI/C3		
POINT_ON_SURFACE	point_on_surface	AI/C3		
SEAM_CURVE	seam_curve	AI/C3		

Table 5 (continued)

Application element	AIM element	Source	Rules	Reference path
non_manifold_surface_model_shape_UoF				
SHELL	connected_face_set	AIC3		connected_face_set → face {⇒
shell to face				connected_face_set.cfs_faces[i] → (face_surface advanced_face) (oriented_face)
SUPERICAL_SURFACE	spherical_surface	AIC3		
SURFACE	surface	AIC3		
SURFACE_MODEL	non_manifold_surface_shape_representation	AIC3	1,9	non_manifold_surface_shape_representation =
user_defined_name	mechanical_design_name_assignment.assigned_name	205	1,9	named_item_select mechanical_design_name_assignment. items[] mechanical_design_name_assignment name_assignment name_assignment.assigned_name
SURFACE_MODEL_REPLICA	mapped_item	43	3	mapped_item mapped_item.mapping_source representation_map representation_map. mapped_representation representation ↑ ↑
surface_model_replica_to				
surface_model				

Table 5 (continued)

Application element	AIM element	Source	Rules	Reference path
	non_manifold_surface_model_shape_UoF			
SURFACE_OF_LINEAR_EXTRUSION	surface_of_linear_extrusion	AIC3	1,9 shape_representation non_manifold_surface_shape_representation ⇒	
SURFACE_OF_REVOLUTION	surface_of_revolution	AIC3		
SURFACE_REPLICA	surface_replica	AIC3	surface_replica. surface_replica. parent_surface surface →	
surface_replica_to_surface				
SWEEP_SURFACE	swept_surface	AIC3		
TOPOLOGICAL_ELEMENT	topological_representation_item	AIC3	11	
user_defined_name	mechanical_design_name_assignment.assigned_name	205	topological_representation_item = named_item_select mechanical_design_name_assignment. items[i] mechanical_design_name_assignment name_assignment name_assignment.assigned_name ←	
TOPOLOGY_SURFACE_MODEL	shell_based_surface_model	AIC3		
TOROIDAL_SURFACE	toroidal_surface	AIC3		

Table 5 (concluded)

Application element	AIM element	Source	Rules	Reference path
	non_manifold_surface_model_shape_UoF			
TRANSFORMATION	cartesian_transformation_operator_3d OR (mapped_item.mapping_target AND mapped_item.mapping_sourceRepresentation_map. mapping_origin)	AIC3 43 43 43	3 3 3	
UNBOUNDED_CURVE	line	AIC3		
VERTEX	vertex_point	AIC3		

Table 6 – Mapping table for product\_structure UoF

Application element	AIM element	Source	Rules	Reference path
product.structure				
ASSEMBLY	product_definition_relationship. relating_product_definition	AIC4	8	{assembly_component_usage product_definition_usage product_definition_relationship} product_definition_relationship. relating_product_definition
coordinate_system .context_of_items	representation.context_of_items	43		product_definition_relationship. relating_product_definition product_definition_relationship characterized_product_definition product_definition_shape = shape_definition shape_definition_representation. representation_of shape_definition_representation shape_definition_representation. representation_model shape_representation representation representation.context_of_items
user_defined_name .name	product.name	AIC4	6	product_definition_relationship. relating_product_definition product_definition_version product_version product_version.of_product product

Table 6 (continued)

Application element	AIM element	Source	Rules	Reference path
product_structure				
assembly to assembly		5,17	{product.frame_of_reference product.context mechanical_context} product.name	↑ ⇒
		8	product_definition.relationship. relating_product_definition product_definition.relationship product_definition.usage assembly_component_usage	↑ ↑ ↑
assembly to product		6 7,18	product_definition.relationship.product_- definition product_definition {product_definition.frame_of_reference product_definition.context design_context} product_definition.version	↑ ↑ ↑ ↑
PART user_defined_name	product_definition.shape product_definition.shape.name	41 41	product_definition.shape product_definition.shape.definition characterized_product_definition product_definition product_definition.relationship. relating_product_definition	↑ = ↓
part to assembly		6		

Table 6 (continued)

Application element	AIM element	Source	Rules	Reference path
	product_structure			
part to product		6	product_definition_shape product_definition_shape.definition → characterized_product_definition = product_definition product_definition.version → product_version	
part to shape_representation			product_definition_shape shape_definition shape_definition_representation. representation_of shape_definition_representation shape_definition_representation. representation_model shape_representation →	
PRODUCT	product_version	AIC4 4.3	product_version product_definition.version product_definition characterized_product_definition ↓ product_definition_shape shape_definition shape_definition_representation. representation_of shape_definition_representation shape_definition_representation. representation_model shape_representation representation.context_of_items	1,9 6 ↓ = ↓ = ↓ = ↑ ↔

Table 6 (continued)

Application element	AIM element	Source	Rules	Reference path
	product_structure			
user_defined_name .name	product.name	AIC4	product_version product_version.of_product product.product.name	→
version_and_id	product_version.id AND product.id	AIC4 AIC4		
SHAPE_REPRESENTATION global_unit	shape_representation global_unit_assigned_context.units	41 41	1,9 shape_representation representation.context_of_items representation.context global_unit_assigned_context global_unit_assigned_context.units[i] unit	↔ → ⇒ = ⇒ → ⇒
		4,14	named_unit ([length_unit] [si_unit]) ([plane_angle_unit] [conversion_based_unit]{ conversion_based_unit.conv_factor measure_with_unit plane_angle_measure_with_unit}))	

Table 6 (continued)

Application element	AIM element	Source	Rules	Reference path
product_structure				
shape_functionality	shape_aspect	41	1,9	shape_representation shape_definition_representation.representation_model shape_definition_representation shape_definition_representation_of shape_definition shape_aspect
TRANSFORMATION	transformation	43		{transformation functionally_defined_transformation cartesian_transformation_operator }
	OR			= ⇒
	mapped_item	43	3	(transformation representation_relationship_with_transformation. transformation_operator representation_relationship_with_transformation representation_relationship representation_relationship.rep_2 (mapped_item representation.item representation.items[i]) representation shape_representation
transformation to assembly				↓ ↔ →) ↓ ↓ ↑ ↓

Table 6 (concluded)

Application element	AIM element	Source	Rules	Reference path
	product_structure			
				shape_definition_representation.representation_of shape_definition product_definition_shape product_definition_shape.definition characterized_product_definition product_definition_relationship product_definition_relationship.related_product_definition
8				= = = = ↓ → → → → representation.relationship_with_transformation. transformation_operator representation.relationship_with_transformation representation.relationship representation.relationship.rep_2 (mapped_item representation.item representation.items[i]) representation shape_representation
				↑ ↓ ↑ ↓ ↑ shape_definition_representation.representation_model shape_definition_representation shape_definition_representation.representation_of shape_definition product_definition_shape
				= = = =
			1,9	

Table 7 – Mapping table for surface\_basic\_visual\_presentation\_UoF

Application element	AIM element	Source	Rules	Reference path
	surface_basic_visual_presentation_UoF			
3D_PROJECTION	camera_model_d3	AIC5		
3D_projection_to_screen_image				<p>camera_model_d3      camera_model      camera_usage.projection (DER)      camera_usage      camera_image.source (DER)      camera_image      geometric_representation.item      representation_item      representation.items[i]      {representation        presentation_representation        product_data_representation_view        product_data_representation_view_with_hllhsr}      {representation_representation        representation_map.mapped_representation        representation_map        mapped_item.mapping_source        mapped_item        representation.items[i]        {representation          representation_representation          presentation_view          area_or_view          background_colour.presentation          background_colour          colour}</p>

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
	surface_basic_visual_representation_UoF			
		<p>representation_map.mapped_representation colour_specification colour_rgb}</p> <p>presentation_size_assignment_select presentation_size_unit}</p> <p>presentation_representation representation}</p> <p>representation_map</p> <p>mapped_item.mapping_source mapped_item</p> <p>representation.items[i] {representation presentation_area}</p> <p>area_or_view background.colour.presentation background.colour colour colour_specification colour_rgb}</p> <p>presentation_size_assignment_select presentation_size_unit}</p> <p>mechanical_design_presentation.area</p>	<p>↑ {= ↓ {↓ ↓ ↓ ↓ ↓ ↑ {= ↓ {↓ ↑ ↑ ↑ ↑ }</p> <p>16</p>	

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
	surface_basic_visual_presentation_UoF			
3D_projection_to_surface_model				<pre> camera_model_d3 camera_usage camera_usage.projection (DER) camera_usage representation_map representation_map.mapped_representation representation_map representation representation mechanical_design_presentation_representation shape_representation (geometrically_boundedsurface_ shape_representation) (manifold_surface_shape_representation) (non_manifold_surface_shape_representation) </pre>
1,9				<pre> AIC5 annotation_text_occurrence annotation_text_occurrence annotation_text_occurrence annotation_text_occurrence styled_item {styled_item.styles[i]} presentation_style_select pre_defined_presentation_style mechanical_design_pre_defined_text_style styled_item.item representation_item geometric_representation_item annotation_text annotation_text.text_source (DER) </pre>

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
<b>surface_basic_visual_representation_UoF</b>				

annotation_text_map	annotation_text_map	→		
text_string_representation	text_map.text_string (DER)	→		
text_string_representation.strings[i]	→	=		
text_or_character	=	↔		
(annotation_text)	(text_literal_mapped_item	→		
(text_literal_text)	mapped_item	→		
mapped_item	mapped_item.mapping_source	→		
representation_map	representation_map.mapped_representation	→		
text_literal_symbol	text_literal_symbol	→		
text_literal_symbol.font	text_literal_symbol.font	→		
font_select	font_select	→	=	↑
pre_defined_text_font	pre_defined_text_font	↓	=	↑
mechanical_design.pre_defined_text_font)	representation.items[i]	↓	↑	⇒}
representation	{representation	↑	⇒}	↔
presentation_representation	presentation_representation	↓	↔	↔
view_dependent_annotation_representation	view_dependent_annotation_representation	↓	↔	↔
{presentation_representation	{presentation_representation	↓	↓	↓
representation}	representation}	↓	↓	↓
representation_map	representation_map.mapped_representation	↓	↓	↓
mapped_item	mapped_item.mapping_source	↓	↓	↑
mapped_item	representation.items[i]	↓	↑	⇒}
representation	presentation_representation	↓	↑	⇒}

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
surface_basic_visual_presentation_UoF				
				<p>↓ {presentation_view representation}</p> <p>↓ representation_map.mapped_representation</p> <p>↓ representation_map</p> <p>↓ mapped_item.mapping_source</p> <p>↓ mapped_item</p> <p>↓ representation.items[i]</p> <p>↑ {representation presentation_representation presentation_area mechanical_design_presentation_area}</p> <p>↑ annotation_text_occurrence</p> <p>↑ annotation_occurrence</p> <p>↑ styled_item</p> <p>↑ representation_item</p> <p>↑ geometric_representation_item</p> <p>↑ annotation_text</p> <p>↑ mapped_item</p> <p>↑ {[mapped_item.mapping_target] [mapped_item.mapping_source]}</p> <p>↑ representation_map</p> <p>↑ representation_map.mapping_origin</p> <p>↑ representation_item</p> <p>↑ geometric_representation_item</p> <p>↑ placement</p> <p>↑ axis2_placement_2d}</p>
annotation_text to transformation	16			
	10			

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
surface_basic_visual_presentation_UoF				
CURVE_APPEARANCE	curve_style	AIC5		curve_style curve_style.curve.colour colour colour_specification
colour	colour.rgb	AIC5		colour.rgb ↑ ↑↑ ↑↑
curve_font	curve_style_font	AIC5		curve_style curve_font curve_font_or_scaled_curve_font_select curve_style_font_select curve_style_font curve_width
curve_width	curve_style.curve_width	AIC5		curve_style curve_style curve_style curve_style curve_style curve_width
curve_appearance_to_curve				presentation_style_select presentation_style_assignment.styles[i] presentation_style_assignment styled_item.styles[i] styled_item 10 representation_dependent_styled_item styled_item.item representation_item geometric_representation_item curve
curve_appearance_to_edge				curve_style presentation_style_select presentation_style_assignment.styles[i] presentation_style_assignment ↓

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
surface_basic_visual_presentation_UoF				
		1.0	styled_item.styles[i]	{⇒ representation_dependent_styled_item} ↑ styling_item representation_item topological_representation_item ⇒ edge
		11		
LAYER	presentation_layer_assignment	AIC5		
user_defined_name	mechanical_design_name_assignment.assigned_name	205		{← presentation_layer_assignment presentation_layer_usage_assignment} presentation_layer_assignment. layered_representation[i] layered_item geometric_representation_item named_item_select mechanical_design_name_assignment. items[i] mechanical_design_name_assignment name_assignment name_assignment.assigned_name ⇒

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
surface_basic_visual_presentation_UoF				
layer to assembly				<p>presentation_layer_assignment presentation_layer_assignment. layered_representation[i]</p> <p>layered_item representation shape_representation shape_definition_representation.</p> <p>representation_model shape_definition_representation shape_definition_representation. representation_of</p> <p>shape_definition product_definition_shape product_definition_shape. definition</p> <p>characterized_product_definition product_definition_relationship product_definition_relationship. relating_product_definition</p>
	1,9			<p>presentation_layer_assignment presentation_layer_assignment. layered_representation[i]</p> <p>layered_item geometric_representation_item</p>
layer to geometric_element				

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
surface_basic_visual_presentation_UoF				
layer to layer				<p>presentation_layer_assignment layered_representation[i]</p> <p>→ ↓</p> <p>layered_item</p> <p>presentation_layer_assignment. layered_representation[i]</p> <p>presentation_layer_assignment</p>
layer to part			1,9	<p>presentation_layer_assignment presentation_layer_assignment</p> <p>layered_representation[i]</p> <p>layered_item</p> <p>representation</p> <p>shape_representation shape_definition_representation.</p> <p>shape_definition_model</p> <p>shape_definition_representation</p> <p>shape_definition_representation. representation_of</p> <p>shape_definition</p> <p>product_definition_shape</p> <p>→ = ↑ ↓</p>

Table 7 (continued)

Application element	AIM element	Source	Rules	bf Reference path
surface_basics_visual_presentation_UoF				
layer to presentation_appearance				presentation.layer.assignment. presentation.layer.assignment[i] layered_representation[i]
layer to product				presentation.layer.assignment presentation.layer.assignment[i] layered_representation[i] representation

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
surface_basic_visual_presentation_UoF				
		1,9	$\downarrow$ shape_representation shape_definition_representation. representation_model shape_definition_representation shape_definition_representation. representation_of shape_definition product_definition_shape product_definition_shape_definition characterized_product_definition [product_definition] [product_definition_relationship] product_definition_relationship. relating_product_definition product_definition product_definition_version product_version	$\uparrow = \uparrow$ $\uparrow = \uparrow$
Layer to surface_model				presentation_layer_assignment presentation_layer_assignment. layered_representation[i] layered_item representation shape_representation (geometrically_boundedsurface_ shape_representation) (manifold_surface_shape_representation) (non_manifold_surface_shape_representation)

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
surface_basic_visual_presentation_UoF				
layer to topological_element				presentation_layer_assignment presentation_layer_assignment. layered_representation[i] → =
POINT_APPEARANCE	point_style	AIC5	11	[geometric_representation_item] [topological_representation_item]
colour	colour_rgb	AIC5		point_style point_style.marker.colour colour colour_specification colour.rgb → ⇒ ⇒
marker	marker_type	AIC5		point_style point_style.marker marker.select marker_type → =
marker_size	point_style.marker_size	AIC5		point_style point_style.marker_size point_style =
point_appearance to point				point_style_select presentation_style_select presentation_style_assignment.styles[i] presentation_style_assignment styled_item.styles[i] →

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
surface_basic_visual_presentation_UoF				
		10	styled_item representation_dependent_styled_item styled_item.item representation_item geometric_representation_item point	{⇒ ↑ ⇒ ⇒}
PRESENTATION_APPEARANCE	presentation_style_assignment.styles[i]	AIC5		
presentation_appearance to shell			presentation_style_assignment.styles[i] presentation_style_assignment styled_item.styles[i] styled_item representation_dependent_styled_item styled_item.item representation_item topological_representation_item connected_face_set	↓ {⇒ ↑ ⇒ ⇒}
presentation_appearance to surface_model			presentation_style_assignment. styles[i] presentation_style_assignment styled_item.styles[i]	↓

Table 7 (continued)

Application element	AIM element	Source	Rules	Reference path
	surface_basic_visual_presentation_UoF			
		10	styled_item	{ $\Rightarrow$ representation.dependent_styled_item} $\rightarrow$ styled_item.item representation.item representation.items[i] $\downarrow$ representation $\Rightarrow$ shape_representation (manifold_surface_shape_representation) (non_manifold_surface_ shape_representation) (geometrically_boundedsurface_shape_representation)
1,9				
SCREEN_IMAGE	mechanical_design_presentation_area	AIC5		
SURFACE_APPEARANCE colour	surface_style_usage colour_rgb	AIC5	surface_style_usage surface_style_usage.style surface_side_style_select surface_side_style surface_side_style.styles[i] surface_style_element_select surface_style_rendering surface_style_rendering.surface_colour colour colour_specification colour_rgb	$\rightarrow$ $\Rightarrow$ $\Rightarrow$ $\Rightarrow$ $\Rightarrow$ $\Rightarrow$ $\Rightarrow$ $\Rightarrow$ $\Rightarrow$ $\Rightarrow$

Table 7 (continued)

Application element	AIM element	Source	Rules	bf Reference path
		surface_basic_visual_presentation_U_OF		
grid_indicator	surface_style_control_grid	AIC5		surface_style_usage surface_style_usage.style surface_side_style_select surface_side_style surface_side_style.styles[i] surface_style_element_select (surface_style_boundary) (surface_style_control_grid) (surface_style_parameter_line) (surface_style_segmentation_curve) (surface_style_silhouette)
shading_method	surface_style_rendering_rendering_method	AIC5		surface_style_usage surface_style_usage.style surface_side_style_select surface_side_style surface_side_style.styles[i] surface_style_element_select surface_style_rendering surface_style_rendering.rendering_method

Table 7 (concluded)

Application element	AIM element	Source	Rules	Reference path
surface_basic_visual_presentation_UoF				
surface_appearance_to_face				surface_style_usage presentation_style_select presentation_style_assignment styled_item.styles[i] styled_item {⇒ representation_dependent_styled_item} styled_item.item representation_item topological_representation_item face
		10		
		11		
surface_appearance_to_surface				surface_style_usage presentation_style_select presentation_style_assignment styled_item.styles[i] styled_item {⇒ representation_dependent_styled_item} styled_item.item representation_item geometric_representation_item surface
		10		

### Rules applied to table

The following rules are referenced in the preceding tables:

- 1.: alternative\_surface\_shape\_representations
- 2.: dependent\_instantiation\_of\_geometry
- 3.: dependent\_instantiation\_of\_mapped\_item
- 4.: dependent\_instantiation\_of\_named\_unit
- 5.: dependent\_instantiation\_of\_product\_context
- 6.: dependent\_instantiation\_of\_product\_definition
- 7.: dependent\_instantiation\_of\_product\_definition\_context
- 8.: dependent\_instantiation\_of\_product\_definition\_relationship
- 9.: dependent\_instantiation\_of\_shape\_representation
- 10.: dependent\_instantiation\_of\_styled\_item
- 11.: dependent\_instantiation\_of\_topology
- 12.: global\_units\_in\_geometric\_representation\_context
- 13.: global\_units\_required
- 14.: illegal\_complex\_named\_units
- 15.: no\_complex\_geometric\_representation\_item
- 16.: presentation\_area\_mechanical\_design
- 17.: product\_context\_mechanical
- 18.: product\_definition\_context\_design
- 19.: three\_dimensional\_geometry\_mainly

## 5.2 AIM EXPRESS short listing

This clause specifies the *EXPRESS* schema that uses elements from the integrated resources and the AICs and contains the types, entity specializations, rules, and functions that are specific to this part of ISO 10303. This clause also specifies modifications to the textual material for

the constructs that are imported from the AICs and the integrated resources. The definitions and *EXPRESS* provided in the integrated resources for constructs used in the AIM may include select list items and subtypes which are not imported into the AIM. Requirements stated in the integrated resources which refer to such items and subtypes apply exclusively to those items which are imported into the AIM.

```
*)  
SCHEMA mechanical_design_surface_schema;  
  
USE FROM product_definition_schema  
  (product_related_product_category);  
  -- ISO 10303-41  
  
USE FROM product_property_representation_schema  
  (shape_representation,  
   shape_definition_representation);  
  -- ISO 10303-41  
  
USE FROM product_property_definition_schema  
  (shape_aspect,  
   product_definition_shape);  
  -- ISO 10303-41  
  
USE FROM management_resources_schema  
  (group_assignment,  
   name_assignment);  
  -- ISO 10303-41  
  
USE FROM measure_schema  
  (length_unit,  
   plane_angle_unit,  
   named_unit,  
   si_unit,  
   conversion_based_unit,  
   plane_angle_measure_with_unit,  
   global_unit_assigned_context);  
  -- ISO 10303-41  
  
USE FROM topology_schema  
  (topological_representation_item);  
  -- ISO 10303-42  
  
USE FROM representation_schema  
  (representation_relationship_with_transformation,  
   geometric_representation_item,  
   representation_context,
```

```
geometric_representation_context);
-- ISO 10303-43
```

```
USE FROM presentation_organization_schema
(presentation_layer_assignment);
-- ISO 10303-46
```

```
USE FROM aic_gbnd_surf;
```

```
USE FROM aic_non_mfld_surf;
```

```
USE FROM aic_mfld_surf;
```

```
USE FROM aic_mech_dsgn_ctxt;
```

```
USE FROM aic_mech_dsgn_pres;
```

(\*

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

product_property_representation_schema	ISO 10303-41
management_resources_schema	ISO 10303-41
measure_schema	ISO 10303-41
topology_schema	ISO 10303-42
representation_schema	ISO 10303-43

2 – One further AIC is used indirectly in this AP; the **aic\_mfld\_surf** and **aic\_non\_mfld\_surf** use the **aic\_top\_bnd\_surf**.

3 – Part numbers are not yet assigned to the AIC documents

## 5.2.1 Mechanical design using surface representation type definitions

### 5.2.1.1 grouped\_item\_select

The **grouped\_item\_select** provides a mechanism to identify shapes that may be assigned to a **group**.

EXPRESS specification:

\*)

```
TYPE grouped_item_select = SELECT
(geometric_representation_item,
 topological_representation_item,
```

```

    shape_representation);
END_TYPE;
(*

```

### 5.2.1.2 named\_item\_select

The **named\_item\_select** provides a mechanism to identify shapes and layers that may be assigned to a name.

EXPRESS specification:

```

*)
TYPE named_item_select = SELECT
  (geometric_representation_item,
   topological_representation_item,
   shape_representation,
   presentation_layer_assignment);
END_TYPE;
(*

```

## 5.2.2 Mechanical design using surface representation entities

### 5.2.2.1 Mechanical design using surface representation entity definitions

#### 5.2.2.1.1 mechanical\_design\_group\_assignment

A **mechanical\_design\_group\_assignment** identifies representations of shape that belong to a **group**. Only entities of type **grouped\_item\_select** may become elements of **groups**.

EXPRESS specification:

```

*)
ENTITY mechanical_design_group_assignment
  SUBTYPE OF (group_assignment);
  items : SET [1:?] OF grouped_item_select;
END_ENTITY;
(*

```

Attribute definitions:

**items:** the entity instances that are assigned to a single **group**.

**SELF\group\_assignment.assigned\_group.name:** the user-defined name for the **group** that **items** are assigned to.

#### 5.2.2.1.2 mechanical\_design\_name\_assignment

A **mechanical\_design\_name\_assignment** identifies one or several representations of shape and layers that are assigned one name.

EXPRESS specification:

```
*)  
ENTITY mechanical_design_name_assignment  
  SUBTYPE OF (name_assignment);  
  items : SET [1:?] OF named_item_select;  
END_ENTITY;  
(*
```

Attribute definitions:

**items:** the items to be named. **items** may be geometric or topological entities, **shape-representations**, or **presentation\_layer\_assignments**.

**SELF\name\_assignment.name:** the user-defined name for **items**.

## 5.2.2.2 Mechanical design using surface representation imported entity modifications

### 5.2.2.2.1 geometric\_representation\_context

The base definition of the **geometric\_representation\_context** entity is given in ISO 10303-43: 4.4.10. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **geometric\_representation\_context** entity:

- **global\_units\_in\_geometric\_representation\_context** (See ??).

### 5.2.2.2.2 geometric\_representation\_item

The base definition of the **geometric\_representation\_item** entity is given in ISO 10303-43: 4.4.11. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **geometric\_representation\_item** entity:

- **dependent\_instantiation\_of\_geometry** (See ??);
- **no\_complex\_geometric\_representation\_item** (See ??);
- **three\_dimensional\_geometry\_mainly** (See ??).

### 5.2.2.2.3 global\_unit\_assigned\_context

The base definition of the **global\_unit\_assigned\_context** entity is given in ISO 10303-41: 23.4.20. The following modifications apply to this part of ISO 10303.

The following additional informal propositions apply:

**IP1:** Any entity instance containing a **length\_measure** as part of its definition shall occur in a **global\_unit\_assigned\_context**.

**IP2:** Any entity instance containing a **plane\_angle\_measure** as part of its definition shall occur in a **global\_unit\_assigned\_context** for which a **plane\_angle\_unit** is defined.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **global\_unit\_assigned\_context** entity:

- **global\_units\_required** (See ??).

#### 5.2.2.2.4 mapped\_item

The base definition of the **mapped\_item** entity is given in ISO 10303-43: 4.4.9. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **mapped\_item** entity:

- **dependent\_instantiation\_of\_mapped\_item** (See ??).

#### 5.2.2.2.5 named\_unit

The base definition of the **named\_unit** entity is given in ISO 10303-41: 23.4.1. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **named\_unit** entity:

- **dependent\_instantiation\_of\_named\_unit** (See ??);
- **illegal\_complex\_named\_units** (See ??).

#### 5.2.2.2.6 presentation\_area

The base definition of the **presentation\_area** entity is given in ISO 10303-46: 4.4.3. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **presentation\_area** entity:

- **presentation\_area\_mechanical\_design** (See ??).

#### 5.2.2.2.7 product\_context

The base definition of the **product\_context** entity is given in ISO 10303-41: 5.3.3. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product\_context** entity:

- dependent\_instantiation\_of\_product\_context (See ??);
- product\_context\_mechanical (See ??).

### **5.2.2.2.8 product\_definition**

The base definition of the **product\_definition** entity is given in ISO 10303-41: 6.4.7. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product\_definition** entity:

- dependent\_instantiation\_of\_product\_definition (See ??).

### **5.2.2.2.9 product\_definition\_context**

The base definition of the **product\_definition\_context** entity is given in ISO 10303-41: 5.3.4. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product\_definition\_context** entity:

- dependent\_instantiation\_of\_product\_definition\_context (See ??);
- product\_definition\_context\_design (See ??).

### **5.2.2.2.10 product\_definition\_relationship**

The base definition of the **product\_definition\_relationship** entity is given in ISO 10303-41: 6.4.9. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product\_definition\_relationship** entity:

- dependent\_instantiation\_of\_product\_definition\_relationship (See ??).

### **5.2.2.2.11 product\_related\_product\_category**

The base definition of the **product\_related\_product\_category** entity is given in ISO 10303-41: 6.4.3. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product\_related\_product\_category** entity:

- dependent\_instantiation\_of\_product\_related\_product\_category (See ??).

### **5.2.2.2.12 product\_version**

The base definition of the **product\_version** entity is given in ISO 10303-41: 6.4.5. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **product\_version** entity:

- dependent\_instantiation\_of\_product\_version (See ??).

### **5.2.2.2.13 shape\_representation**

The base definition of the **shape\_representation** entity is given in ISO 10303-41: 8.3.1. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **shape\_representation** entity:

- alternative\_surface\_shape\_representations (See ??);
- dependent\_instantiation\_of\_shape\_representation (See ??).

### **5.2.2.2.14 styled\_item**

The base definition of the **styled\_item** entity is given in ISO 10303-46: 6.4.1. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **styled\_item** entity:

- dependent\_instantiation\_of\_styled\_item (See ??).

### **5.2.2.2.15 topological\_representation\_item**

The base definition of the **topological\_representation\_item** entity is given in ISO 10303-42: 5.4.1. The following modifications apply to this part of ISO 10303.

Associated global rules:

The following global rules defined in this part of ISO 10303 apply to the **topological\_representation\_item** entity:

- dependent\_instantiation\_of\_topology (See ??).

### 5.2.3 Mechanical design using surface representation rule definitions

#### 5.2.3.1 alternative\_surface\_shape\_representations

The **alternative\_surface\_shape\_representations** rule ensures that any instance of a **shape\_representation** conforms to the specification of one of:

- a **geometrically\_boundedsurface\_shape\_representation**;
- a **manifold\_surface\_shape\_representation**;
- a **non\_manifold\_surface\_shape\_representation**.

EXPRESS specification:

```
*)  
RULE alternative_surface_shape_representations FOR (shape_representation);  
WHERE  
    WR1 : SIZEOF (QUERY (sr <* shape_representation |  
        NOT (SIZEOF ([ 'MECHANICAL DESIGN_SURFACE_SCHEMA.' +  
            'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION',  
            'MECHANICAL DESIGN_SURFACE_SCHEMA.' +  
            'NON_MANIFOLD_SURFACE_SHAPE REPRESENTATION',  
            'MECHANICAL DESIGN_SURFACE_SCHEMA.' +  
            'MANIFOLD_SURFACE_SHAPE_REPRESENTATION'] * TYPEOF (sr))  
        = 1))) = 0;  
END_RULE;  
(*
```

Argument definitions:

**shape\_representation:** identifies the set of all instances of **shape\_representation** entities to which the rule is applied.

Formal propositions:

**WR1:** A **shape\_representation** shall consist of exactly one of the three alternative surface models, i.e. shall be either one **geometrically\_boundedsurface\_shape\_representation** or one **non\_manifold\_surface\_shape\_representation** or one **manifold\_surface\_shape\_representation**.

#### 5.2.3.2 dependent\_instantiation\_of\_geometry

This rule ensures that any instance of a **geometric\_representation\_item** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)  
RULE dependent_instantiation_of_geometry FOR (geometric_representation_item);  
WHERE  
    WR1 : SIZEOF ( QUERY (gri <* geometric_representation_item |
```

```

    NOT (SIZEOF (USEDIN (gri,'')) > 0 ))) = 0;
END_RULE;
(*

```

Argument definitions:

**geometric\_representation\_item:** identifies the set of all instances of **geometric\_representation\_item**.

Formal propositions:

**WR1:** Any instance of a **geometric\_representation\_item** shall be used in the definition of some other entity.

### 5.2.3.3 dependent\_instantiation\_of\_mapped\_item

This rule ensures that any instance of a **mapped\_item** is used as part of the definition of some other entity.

EXPRESS specification:

```

*)
RULE dependent_instantiation_of_mapped_item FOR (mapped_item);
WHERE
    WR1 : SIZEOF ( QUERY (mi <* mapped_item |
        NOT (SIZEOF (USEDIN (mi,'')) > 0 ))) = 0;
END_RULE;
(*

```

Argument definitions:

**mapped\_item:** identifies the set of all instances of **mapped\_item**.

Formal propositions:

**WR1:** Any instance of a **mapped\_item** shall be used in the definition of some other entity.

### 5.2.3.4 dependent\_instantiation\_of\_mechanical\_design\_presentation\_representation

This rule ensures that any instance of a **mechanical\_design\_presentation\_representation** is used as part of the definition of some other entity.

EXPRESS specification:

```

*)
RULE dependent_instantiation_of_mechanical_design_presentation_representation
    FOR (mechanical_design_presentation_representation);
WHERE
    WR1 : SIZEOF ( QUERY (mdpr <* mechanical_design_presentation_representation |
        NOT (SIZEOF (USEDIN (mdpr,'')) > 0 ))) = 0;
END_RULE;
(*

```

Argument definitions:

**mechanical\_design\_presentation\_representation:** identifies the set of all instances of **mechanical\_design\_presentation\_representation**.

Formal propositions:

**WR1:** Any instance of a **mechanical\_design\_presentation\_representation** shall be used in the definition of some other entity.

### 5.2.3.5 dependent\_instantiation\_of\_named\_unit

This rule ensures that any instance of a **named\_unit** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)  
RULE dependent_instantiation_of_named_unit FOR (named_unit);  
WHERE  
    WR1 : SIZEOF ( QUERY (nmu <* named_unit |  
        NOT (SIZEOF (USEDIN (nmu, '')) > 0 ))) = 0;  
END_RULE;  
(*
```

Argument definitions:

**named\_unit:** identifies the set of all instances of **named\_unit**.

Formal propositions:

**WR1:** Any instance of a **named\_unit** shall be used in the definition of some other entity.

### 5.2.3.6 dependent\_instantiation\_of\_product\_context

This rule ensures that any instance of a **product\_context** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)  
RULE dependent_instantiation_of_product_context FOR (product_context);  
WHERE  
    WR1 : SIZEOF ( QUERY (pc <* product_context |  
        NOT (SIZEOF (USEDIN (pc, '')) > 0 ))) = 0;  
END_RULE;  
(*)
```

Argument definitions:

**product\_context:** identifies the set of all instances of **product\_context**.

Formal propositions:

**WR1:** Any instance of a **product\_context** shall be used in the definition of some other entity.

### 5.2.3.7 dependent\_instantiation\_of\_product\_definition

This rule ensures that any instance of a **product\_definition** is used as part of the definition of some other entity.

EXPRESS specification:

```

*)
RULE dependent_instantiation_of_product_definition FOR (product_definition);
WHERE
  WR1 : SIZEOF ( QUERY (pd <* product_definition |
    NOT (SIZEOF (USEDIN (pd,'')) > 0 ))) = 0;
END_RULE;
(*

```

Argument definitions:

**product\_definition:** identifies the set of all instances of **product\_definition**.

Formal propositions:

**WR1:** Any instance of a **product\_definition** shall be used in the definition of some other entity.

### 5.2.3.8 dependent\_instantiation\_of\_product\_definition\_context

This rule ensures that any instance of a **product\_definition\_context** is used as part of the definition of some other entity.

EXPRESS specification:

```

*)
RULE dependent_instantiation_of_product_definition_context FOR
  (product_definition_context);
WHERE
  WR1 : SIZEOF ( QUERY (pdc <* product_definition_context |
    NOT (SIZEOF (USEDIN (pdc,'')) > 0 ))) = 0;
END_RULE;
(*

```

Argument definitions:

**product\_definition\_context:** identifies the set of all instances of **product\_definition\_context**.

Formal propositions:

**WR1:** Any instance of a **product\_definition\_context** shall be used in the definition of some other entity.

### 5.2.3.9 dependent\_instantiation\_of\_product\_definition\_relationship

This rule ensures that any instance of a **product\_definition\_relationship** is used as part of the definition of some other entity.

EXPRESS specification:

```

*)
RULE dependent_instantiation_of_product_definition_relationship FOR
  (product_definition_relationship);
WHERE
  WR1 : SIZEOF ( QUERY (pdr <* product_definition_relationship |
    NOT (SIZEOF (USEDIN (pdr,'')) > 0 ))) = 0;

```

```
END_RULE;
(*
```

Argument definitions:

**product\_definition\_relationship:** identifies the set of all instances of **product\_definition\_relationship**.

Formal propositions:

**WR1:** Any instance of a **product\_definition\_relationship** shall be used in the definition of some other entity.

### 5.2.3.10 dependent\_instantiation\_of\_product\_related\_product\_category

This rule ensures that any instance of a **product\_related\_product\_category** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_product_related_product_category
  FOR (product_related_product_category);
WHERE
  WR1 : SIZEOF ( QUERY (prpc <* product_related_product_category |
    NOT (SIZEOF (USEDIN (prpc,'')) > 0 ))) = 0;
END_RULE;
(*)
```

Argument definitions:

**product\_related\_product\_category:** identifies the set of all instances of **product\_related\_product\_category**.

Formal propositions:

**WR1:** Any instance of a **product\_related\_product\_category** shall be used in the definition of some other entity.

### 5.2.3.11 dependent\_instantiation\_of\_product\_version

This rule ensures that any instance of a **product\_version** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_product_version FOR (product_version);
WHERE
  WR1 : SIZEOF ( QUERY (pv <* product_version |
    NOT (SIZEOF (USEDIN (pv,'')) > 0 ))) = 0;
END_RULE;
(*)
```

Argument definitions:

**product\_version:** identifies the set of all instances of **product\_version**.

Formal propositions:

**WR1:** Any instance of a **product\_version** shall be used in the definition of some other entity.

### 5.2.3.12 dependent\_instantiation\_of\_shape\_representation

This rule ensures that any instance of a **shape\_representation** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_shape_representation FOR (shape_representation);
WHERE
    WR1 : SIZEOF ( QUERY (sr <* shape_representation |
        NOT (SIZEOF (USEDIN (sr,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

**shape\_representation:** identifies the set of all instances of **shape\_representation**.

Formal propositions:

**WR1:** Any instance of a **shape\_representation** shall be used in the definition of some other entity.

### 5.2.3.13 dependent\_instantiation\_of\_styled\_item

This rule ensures that any instance of a **styled\_item** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_styled_item FOR (styled_item);
WHERE
    WR1 : SIZEOF ( QUERY (si <* styled_item |
        NOT (SIZEOF (USEDIN (si,'')) > 0 ))) = 0;
END_RULE;
(*)
```

Argument definitions:

**styled\_item:** identifies the set of all instances of **styled\_item**.

Formal propositions:

**WR1:** Any instance of a **styled\_item** shall be used in the definition of some other entity.

### 5.2.3.14 dependent\_instantiation\_of\_topology

This rule ensures that any instance of a **topological\_representation\_item** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)  
RULE dependent_instantiation_of_topology FOR (topological_representation_item);  
WHERE  
    WR1 : SIZEOF ( QUERY (tri <* topological_representation_item |  
        NOT (SIZEOF (USEDIN (tri,'')) > 0 ))) = 0;  
END_RULE;  
(*
```

Argument definitions:

**topological\_representation\_item**: identifies the set of all instances of **topological\_representation\_item**.

Formal propositions:

**WR1**: Any instance of a **topological\_representation\_item** shall be used in the definition of some other entity.

### 5.2.3.15 global\_units\_in\_geometric\_representation\_context

The **global\_units\_in\_geometric\_representation\_context** rule requires that each **geometric\_representation\_context** is also a **global\_unit\_assigned\_context**. This implies that in case of cartesian geometry all units shall be defined as global units that are valid within specific contexts.

EXPRESS specification:

```
*)  
RULE global_units_in_geometric_representation_context FOR  
    (geometric_representation_context);  
WHERE  
    WR1 : SIZEOF (QUERY (grc <* geometric_representation_context |  
        NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.' +  
            'GLOBAL_UNIT_ASSIGNED_CONTEXT' IN TYPEOF (grc)))) = 0 ;  
END_RULE;  
(*)
```

Argument definitions:

**geometric\_representation\_context**: identifies the set of all instances of **geometric\_representation\_context** entities to which the rule is applied.

Formal propositions:

**WR1**: The instance of a **geometric\_representation\_context** shall always also be of type **global\_unit\_assigned\_context**. That means that cartesian geometry shall always be assigned default units, whereas parametric geometry that does reference **parameter\_value** such as **pcurve** need not be assigned units.

### 5.2.3.16 global\_units\_required

The **global\_units\_required** rule specifies the units that shall be defined for **global\_unit\_assigned\_context**. Every **global\_unit\_assigned\_context** shall have a maximum of 2 elements

in its set of units, these shall include a unit of length and may also include a unit of plane angle measure.

EXPRESS specification:

```
*)
RULE global_units_required FOR (global_unit_assigned_context);
WHERE
    WR1 : SIZEOF ( QUERY (guac <* global_unit_assigned_context |
        NOT (SIZEOF (guac.units) <= 2))) = 0;
    WR2 : SIZEOF ( QUERY (guac <* global_unit_assigned_context |
        NOT (SIZEOF (QUERY( u <* guac.units |
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.LENGTH_UNIT' IN TYPEOF
            (u)))) = 1 ))) = 0;
    WR3: SIZEOF ( QUERY (guac <* global_unit_assigned_context |
        NOT (SIZEOF (QUERY( u <* guac.units |
            NOT (SIZEOF (QUERY (other_u <* guac.units - u |
                'MECHANICAL_DESIGN_SURFACE_SCHEMA.LENGTH_UNIT' IN TYPEOF(u))
                AND
                'MECHANICAL_DESIGN_SURFACE_SCHEMA.PLANE_ANGLE_UNIT' IN
                TYPEOF (other_u)))) = 0))) = 0));
END_RULE;
(*
```

Argument definitions:

**global\_unit\_assigned\_context:** identifies the set of all instances of **global\_unit\_assigned\_context**.

Formal propositions:

**WR1:** For any instance of a **global\_unit\_assigned\_context** the set of units shall contain 2 or less elements.

**WR2:** For any instance of a **global\_unit\_assigned\_context** the set of units shall contain a **length\_unit**.

**WR3:** For any instance of a **global\_unit\_assigned\_context** the set of units shall contain only **length\_units** or **plane\_angle\_units**.

### 5.2.3.17 illegal\_complex\_named\_units

The following rule ensures that a **named\_unit** can not simultaneously be a **length\_unit** and a **plane\_angle\_unit** and that a **named\_unit** can not simultaneously be a **length\_unit** and a **conversion\_based\_unit**.

EXPRESS specification:

```
*)
RULE illegal_complex_named_units FOR (named_unit);
WHERE
    WR1: SIZEOF (QUERY (nu <* named_unit |
        NOT (SIZEOF (TYPEOF(nu) *
            ['MECHANICAL_DESIGN_SURFACE_SCHEMA.LENGTH_UNIT',
```

```

    'MECHANICAL_DESIGN_SURFACE_SCHEMA.PLANE_ANGLE_UNIT']) < 2 )
AND
NOT (SIZEOF (TYPEOF(nu) *
[ 'MECHANICAL_DESIGN_SURFACE_SCHEMA.LENGTH_UNIT',
  'MECHANICAL_DESIGN_SURFACE_SCHEMA.CONVERSION_BASED_UNIT']) < 2 )
)) = 0;
END_RULE;
(*

```

Argument definitions:

**named\_unit:** identifies the set of all instances of **named\_unit** entities to which the rule is applied.

Formal propositions:

**WR1:** The TYPEOF function shall not return two or more of the subtypes in the two lists in this rule for any **named\_unit**. I.e. a **named\_unit** shall not be used as a combination of **length\_unit** and **plane\_angle\_unit**. And a **length\_unit** shall not be based on conversion, but shall be a pure **si\_unit**. Only **plane\_angle\_units** may deviate from **si\_units**.

### 5.2.3.18 no\_complex\_geometric\_representation\_item

The following rule is equivalent to a ONEOF declaration and ensures that no subtype of **geometric\_representation\_item** can simultaneously be of more than one of the subtypes listed in the rule.

EXPRESS specification:

```

*)
RULE no_complex_geometric_representation_item FOR
  (geometric_representation_item);
WHERE
  WR1: SIZEOF (QUERY (gri <* geometric_representation_item |
    NOT (SIZEOF (TYPEOF(gri) *
      [ 'MECHANICAL_DESIGN_SURFACE_SCHEMA.ANNOTATION_TEXT',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.CAMERA_IMAGE',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.CAMERA_MODEL',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.CARTESIAN_TRANSFORMATION_OPERATOR',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.CURVE',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.DIRECTION',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.EDGE_CURVE',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.FACE_BASED_SURFACE_MODEL',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.FACE_SURFACE',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.GEOMETRIC_SET',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.PLACEMENT',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.PLANAR_EXTENT',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.POINT',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.SHELL_BASED_SURFACE_MODEL',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.SURFACE',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.VECTOR',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.VERTEX_POINT']) < 2 ))) = 0;
END_RULE;
(*

```

Argument definitions:

**geometric\_representation\_item:** identifies the set of all instances of **geometric\_representation\_item** entities to which the rule is applied.

Formal propositions:

**WR1:** The TYPEOF function shall not return two or more of the subtypes listed in this rule for any **geometric\_representation\_item**.

### 5.2.3.19 presentation\_area\_mechanical\_design

In the context of this part of ISO 10303 the only valid **presentation\_area** is the subtype **mechanical\_design\_presentation\_area**.

EXPRESS specification:

```
*)
RULE presentation_area_mechanical_design FOR (presentation_area);
WHERE
  WR1 : SIZEOF (QUERY (pa <* presentation_area |
    NOT ('MECHANICAL DESIGN_SURFACE_SCHEMA.' +
      'MECHANICAL DESIGN_PRESENTATION_AREA' IN TYPEOF(pa)))) = 0;
END_RULE;
(*
```

Argument definitions:

**presentation\_area:** identifies the set of all instances of **presentation\_area** entities.

Formal propositions:

**WR1:** A **presentation\_area** shall always be instantiated as its subtype **mechanical\_design\_presentation\_area**.

### 5.2.3.20 product\_context\_mechanical

In the context of this part of ISO 10303 the only valid **product\_context** is the subtype **mechanical\_context**.

EXPRESS specification:

```
*)
RULE product_context_mechanical FOR (product_context);
WHERE
  WR1 : SIZEOF ( QUERY (pc <* product_context |
    NOT ('MECHANICAL DESIGN_SURFACE_SCHEMA.MECHANICAL_CONTEXT'
      IN TYPEOF(pc)))) = 0;
END_RULE;
(*)
```

Argument definitions:

**product\_context:** identifies the set of all instances of **product\_context** entities.

Formal propositions:

**WR1:** Any instance of a **product\_context** shall be of the **mechanical\_context** subtype.

### 5.2.3.21 product\_definition\_context\_design

In the context of this part of ISO 10303 the only valid **product\_definition\_context** is the subtype **design\_context**.

EXPRESS specification:

```
*)  
RULE product_definition_context_design FOR (product_definition_context);  
WHERE  
    WR1 : SIZEOF ( QUERY (pdc <* product_definition_context |  
        NOT ('MECHANICAL DESIGN_SURFACE_SCHEMA.DESIGN_CONTEXT'  
        IN TYPEOF(pc)))) = 0;  
END_RULE;  
(*
```

Argument definitions:

**product\_definition\_context**: identifies the set of all instances of **product\_definition\_context** entities.

Formal propositions:

**WR1**: Any instance of a **product\_definition\_context** shall be of subtype **design\_context**.

### 5.2.3.22 three\_dimensional\_geometry\_mainly

The following rule ensures that all geometry used in the definition of curves and surfaces is three dimensional. Two dimensional geometry may only be used for the visual presentation of two dimensional images or for the definition of curves that are defined in the parameter space of a surface. All geometric representation items have a derived integer attribute dim which gives the dimension of the space in which they are defined. This rule verifies that two dimensional geometry is only used where appropriate.

EXPRESS specification:

```
*)  
RULE three_dimensional_geometry_mainly FOR (geometric_representation_item);  
WHERE  
    WR1: SIZEOF(QUERY(gri <* geometric_representation_item |  
        (gri.dim = 2)  
        AND  
        NOT ((SIZEOF (['MECHANICAL DESIGN_SURFACE_SCHEMA.PLANAR_BOX',  
            'MECHANICAL DESIGN_SURFACE_SCHEMA.ANNOTATION_TEXT'] *  
            TYPEOF(gri)) = 1 )  
        OR  
        (SIZEOF (USEDIN(gri, 'MECHANICAL DESIGN_SURFACE_SCHEMA.' +  
            'DEFINITIONAL REPRESENTATION ITEM.ITEMS[1]') > 0)))) = 0;  
END_RULE;  
(*)
```

Argument definitions:

**geometric\_representation\_item**: identifies the set of all instances of **geometric\_representation\_item** entities to which the rule is applied.

Formal propositions:

**WR1:** Each instance of a **geometric\_representation\_item** shall have a dimension of 3 or be of type **planar\_box** or of type **annotation\_text** or shall be used as **reference\_to\_curve** in a **pcurve**.

\*)

```
END_SCHEMA; -- mechanical_design_surface_schema  
(*
```

ISO/CD 10303-205

## 6 Conformance requirements

Conformance to this part of ISO 10303 includes satisfying the requirements stated in this part, the requirements of the implementation method(s) supported, and the relevant requirements of the normative references.

An implementation shall support at least one of the following implementation methods: ISO 10303-21, ISO 10303-22. Requirements with respect to implementation methods are specified in annex D.

The Protocol Information Conformance Statement (PICS) proforma lists the options or the combination of options that may be included in the implementation. The PICS proforma is provided in annex C.

NOTE – ISO 10303-1205 defines the abstract test suite to be used in the assessment of conformance. ISO 10303-32 describes the conformance assessment process.

This part of ISO 10303 provides for a number of options that may be supported by an implementation. These options have been grouped into the following conformance classes:

Class 1 geometrically bounded surface shape;

Class 2 geometrically bounded surface shape with the presentation of views depicting the shapes;

Class 3 manifold surface shape;

Class 4 manifold surface shape with the presentation of views depicting the shapes;

Class 5 non-manifold surface shape;

Class 6 non-manifold surface shape with the presentation of views depicting the shapes.

Support for a particular conformance class requires support of all the options specified in this class.

Conformance to a specific class is distinguished by the entity **mechanical\_design\_presentation-area** and by the three subtypes of **shape\_representation** listed in table ???. Table ??? does not contain complete listings of required entities for the conformance classes. The four entities mentioned are sufficient for the identification of all members of the respective classes.

Conformance to a particular class requires that all AIM elements defined as part of the items listed in table ?? be supported plus all AIM elements in the **mechanical\_design\_surface-schema** that are not directly or indirectly defined in table ??.

General requirements for all classes are:

- a) The information requirements and relationships of the ARM shall be preserved in the implementation. This includes support for all valid combinations of entities and their attributes. No 'substitution' of entities shall be permitted. Consequently all construct assertions from clause 4 shall be maintained. If there are any conflicts between the requirements in clause 4 and the interpreted model in clause 5, clause 5 takes precedence.

**Table 8 – Conformance classes for this part of ISO 10303**

AIM element	Class					
	1	2	3	4	5	6
geometrically_boundedsurface_shape_representation	X	X				
manifold_surface_shape_representation			X	X		
non_manifold_surface_shape_representation		X			X	X
mechanical_design_presentation_area			X			X

- b) All AIM entities, types, and their associated constraints shall be supported. Treatment of options and default values shall conform to the AIM.
- c) All AIM entities, types, and their associated constraints shall be read and processed by a postprocessor.
- d) Entities not belonging to the AIM shall be excluded from a preprocessor implementation. Entities not specified in the AIM shall not be included in a conforming exchange structure.
- e) An implementation shall satisfy all general requirements of implementation methods as given in the appropriate 20-series classes. This includes the preservation of the AIM during mapping onto the implementation form and conformance to the syntax of the implementation form.
- f) The referenced integrated resources may contain informal propositions that are relevant for conformance to this part of ISO 10303, but which may not be included in this part.

## Annex A

(normative)

### AIM EXPRESS long listing

This file was generated by exppp (an EXPRESS Pretty Printer) written at the National Institute of Standards and Technology by Don Libes, February 19, 1993.

WARNING: If you modify this file and want to save the changes, delete this comment block or else the file will be rewritten the next time exppp processes this schema.

```
*)
SCHEMA mechanical_design_surface_schema;

CONSTANT
  aic_non_manifold_surface_functionality_definition :
    LIST [2:2] OF STRING := ['aic_mfld_surf',
      'the definition of surface models with' +
      ' explicitly defined topological boundaries with characteristics' +
      ' of non-manifolds'];
  aic_topology_bounded_surface_functionality_definition :
    LIST [2:2] OF STRING := ['aic_top_bnd_surf',
      'the definition of surface models with' +
      ' elementary and sculptured geometry and with explicitly' +
      ' defined topological boundaries'];
  aic_geometrically_bounded_surface_functionality_definition :
    LIST [2:2] OF STRING := ['aic_gbnd_surf',
      'the definition of surface models with' +
      ' geometrically defined boundaries'];
  aic_mechanical_design_context_functionality_definition :
    LIST [2:2] OF STRING := ['aic_mech_dsgn_ctxt',
      'product identification and relation for the' +
      ' definition of mechanical products within the design phase of the' +
      ' product life cycle'];
  aic_mechanical_design_presentation_functionality_definition :
    LIST [2:2] OF STRING := ['aic_screen_image',
      'the projection from some mechanical design using' +
      ' either surface or boundary representation to' +
      ' an associated annotated picture on a screen'];
  aic_manifold_surface_functionality_definition :
    LIST [2:2] OF STRING := ['aic_mfld_surf',
      'the definition of surface models with' +
      ' explicitly defined topological boundaries constrained to' +
      ' represent 2-manifolds'];
END_CONSTANT;

TYPE area_or_view = SELECT
```

```
(presentation_area,
 presentation_view);
END_TYPE; -- area_or_view

TYPE axis2_placement = SELECT
 (axis2_placement_2d,
 axis2_placement_3d);
END_TYPE; -- axis2_placement

TYPE bspline_curve_form = ENUMERATION OF
 (elliptic_arc,
 polyline_form,
 parabolic_arc,
 circular_arc,
 unspecified,
 hyperbolic_arc);
END_TYPE; -- bspline_curve_form

TYPE bspline_surface_form = ENUMERATION OF
 (surf_of_linear_extrusion,
 plane_surf,
 generalised_cone,
 toroidal_surf,
 conical_surf,
 spherical_surf,
 unspecified,
 ruled_surf,
 surf_of_revolution,
 cylindrical_surf,
 quadric_surf);
END_TYPE; -- bspline_surface_form

TYPE central_or_parallel = ENUMERATION OF
 (central,
 parallel);
END_TYPE; -- central_or_parallel

TYPE character_spacing_select = SELECT
 (length_measure,
 measure_with_unit);
END_TYPE; -- character_spacing_select

TYPE characterized_product_definition = SELECT
 (product_definition,
 product_definition_relationship);
END_TYPE; -- characterized_product_definition
```

```

TYPE curve_font_or_scaled_curve_font_select = SELECT
  (curve_style_font_select);
END_TYPE; -- curve_font_or_scaled_curve_font_select

TYPE curve_on_surface = SELECT
  (pcurve,
   surface_curve,
   composite_curve_on_surface);
END_TYPE; -- curve_on_surface

TYPE curve_or_annotation_curve_occurrence = SELECT
  (curve);
END_TYPE; -- curve_or_annotation_curve_occurrence

TYPE curve_or_render = SELECT
  (curve_style,
   curve_style_rendering);
END_TYPE; -- curve_or_render

TYPE curve_style_font_select = SELECT
  (curve_style_font);
END_TYPE; -- curve_style_font_select

TYPE dimension_count = INTEGER;
WHERE
  wr1: SELF > 0;
END_TYPE; -- dimension_count

TYPE direction_count_select = SELECT
  (u_direction_count,
   v_direction_count);
END_TYPE; -- direction_count_select

TYPE font_select = SELECT
  (pre_defined_text_font);
END_TYPE; -- font_select

TYPE geometric_set_select = SELECT
  (point,
   curve,
   surface);
END_TYPE; -- geometric_set_select

TYPE grouped_item_select = SELECT
  (geometric_representation_item,

```

```

    topological_representation_item,
    shape_representation);
END_TYPE; -- grouped_item_select

TYPE hiding_or_blanking_select = SELECT
(presentation_area,
 presentation_view,
 product_data_representation_view,
 mapped_item,
 view_dependent_annotation_representation);
END_TYPE; -- hiding_or_blanking_select

TYPE identifier = STRING;
END_TYPE; -- identifier

TYPE knot_type = ENUMERATION OF
(uniform_knots,
 quasi_uniform_knots,
 piecewise_bezier_knots,
 unspecified);
END_TYPE; -- knot_type

TYPE label = STRING;
END_TYPE; -- label

TYPE layered_item = SELECT
(representation,
 geometric_representation_item);
WHERE
wr1: NOT (('MECHANICAL DESIGN_SURFACE_SCHEMA.' +
'REPRESENTATION_ITEM_WITHOUT_STYLE') IN TYPEOF(SELF));
END_TYPE; -- layered_item

TYPE length_measure = REAL;
END_TYPE; -- length_measure

TYPE list_of_reversible_topology_item = LIST [0:?] OF
    reversible_topology_item;
END_TYPE; -- list_of_reversible_topology_item

TYPE marker_select = SELECT
(marker_type);
END_TYPE; -- marker_select

TYPE marker_type = ENUMERATION OF
(dot,

```

```

    plus,
    x,
    square,
    triangle,
    asterisk,
    ring);
END_TYPE; -- marker_type

TYPE measure_value = SELECT
  (length_measure,
   plane_angle_measure,
   parameter_value,
   positive_length_measure);
END_TYPE; -- measure_value

TYPE named_item_select = SELECT
  (geometric_representation_item,
   topological_representation_item,
   shape_representation,
   presentation_layer_assignment);
END_TYPE; -- named_item_select

TYPE null_style = ENUMERATION OF
  (null);
END_TYPE; -- null_style

TYPE parameter_value = REAL;
END_TYPE; -- parameter_value

TYPE pcurve_or_surface = SELECT
  (pcurve,
   surface);
END_TYPE; -- pcurve_or_surface

TYPE plane_angle_measure = REAL;
END_TYPE; -- plane_angle_measure

TYPE positive_length_measure = length_measure;
WHERE
  wr1: SELF > 0;
END_TYPE; -- positive_length_measure

TYPE preferred_surface_curve_representation = ENUMERATION OF
  (pcurve_s2,
   pcurve_s1,
   curve_3d);

```

```
END_TYPE; -- preferred_surface_curve_representation

TYPE presentable_text = STRING;
END_TYPE; -- presentable_text

TYPE presentation_representation_select = SELECT
  (presentation_representation);
END_TYPE; -- presentation_representation_select

TYPE presentation_size_assignment_select = SELECT
  (presentation_area,
   presentation_view);
END_TYPE; -- presentation_size_assignment_select

TYPE presentation_style_select = SELECT
  (pre_defined_presentation_style,
   point_style,
   curve_style,
   surface_style_usage,
   null_style);
END_TYPE; -- presentation_style_select

TYPE reversible_topology = SELECT
  (reversible_topology_item,
   list_of_reversible_topology_item,
   set_of_reversible_topology_item);
END_TYPE; -- reversible_topology

TYPE reversible_topology_item = SELECT
  (edge,
   path,
   face,
   face_bound,
   closed_shell,
   open_shell);
END_TYPE; -- reversible_topology_item

TYPE set_of_reversible_topology_item = SET [0:?] OF
  reversible_topology_item;
END_TYPE; -- set_of_reversible_topology_item

TYPE shading_curve_method = ENUMERATION OF
  (constant_colour,
   linear_colour);
END_TYPE; -- shading_curve_method
```

```

TYPE shading_surface_method = ENUMERATION OF
  (colour_shading,
   constant_shading,
   dot_shading,
   normal_shading);
END_TYPE; -- shading_surface_method

TYPE shape_definition = SELECT
  (product_definition_shape,
   shape_aspect);
END_TYPE; -- shape_definition

TYPE shell = SELECT
  (open_shell,
   closed_shell);
END_TYPE; -- shell

TYPE si_prefix = ENUMERATION OF
  (exa,
   pico,
   mega,
   femto,
   atto,
   centi,
   nano,
   hecto,
   micro,
   tera,
   giga,
   milli,
   peta,
   deci,
   kilo,
   deca);
END_TYPE; -- si_prefix

TYPE si_unit_name = ENUMERATION OF
  (hertz,
   degree_celsius,
   siemens,
   sievert,
   lux,
   watt,
   ohm,
   second,
   becquerel,

```

```
pascal,
henry,
tesla,
volt,
joule,
kelvin,
ampere,
gram,
steradian,
mole,
lumen,
gray,
candela,
farad,
radian,
newton,
metre,
weber,
coulomb);
END_TYPE; -- si_unit_name

TYPE size_select = SELECT
(positive_length_measure,
 measure_with_unit);
END_TYPE; -- size_select

TYPE style_context_select = SELECT
(presentation_area,
 presentation_view,
 product_data_representation_view,
 presentation_layer_usage);
END_TYPE; -- style_context_select

TYPE surface_model = SELECT
(shell_based_surface_model,
 face_based_surface_model);
END_TYPE; -- surface_model

TYPE surface_side = ENUMERATION OF
(negative,
 positive,
 both);
END_TYPE; -- surface_side

TYPE surface_side_style_select = SELECT
(surface_side_style);
```

```

END_TYPE; -- surface_side_style_select

TYPE surface_style_element_select = SELECT
  (surface_style_rendering,
   surface_style_boundary,
   surface_style_silhouette,
   surface_style_segmentation_curve,
   surface_style_control_grid,
   surface_style_parameter_line);
END_TYPE; -- surface_style_element_select

TYPE text = STRING;
END_TYPE; -- text

TYPE text_alignment = label;
END_TYPE; -- text_alignment

TYPE text_or_character = SELECT
  (annotation_text,
   text_literal_mapped_item);
END_TYPE; -- text_or_character

TYPE text_path = ENUMERATION OF
  (up,
   right,
   down,
   left);
END_TYPE; -- text_path

TYPE transformation = SELECT
  (functionally_defined_transformation);
END_TYPE; -- transformation

TYPE transition_code = ENUMERATION OF
  (discontinuous,
   cont_same_gradient_same_curvature,
   cont_same_gradient,
   continuous);
END_TYPE; -- transition_code

TYPE trimming_preference = ENUMERATION OF
  (parameter,
   unspecified,
   cartesian);
END_TYPE; -- trimming_preference

```

```

TYPE trimming_select = SELECT
  (cartesian_point,
   parameter_value);
END_TYPE; -- trimming_select

TYPE u_direction_count = INTEGER;
WHERE
  WR1: SELF > 1;
END_TYPE;

TYPE unit = SELECT
  (named_unit);
END_TYPE; -- unit

TYPE v_direction_count = INTEGER;
WHERE
  WR1: SELF > 1;
END_TYPE;

TYPE vector_or_direction = SELECT
  (vector,
   direction);
END_TYPE; -- vector_or_direction

TYPE visibility_context_select = SELECT
  (presentation_representation,
   presentation_layer_usage);
END_TYPE; -- visibility_context_select

TYPE year_number = INTEGER;
END_TYPE; -- year_number

ENTITY advanced_face
  SUBTYPE OF (face_surface);
  aic_functionality : LIST [2:2] OF STRING;
WHERE
  WR1 : aic_functionality =
    aic_topology_bounded_surface_functionality_definition;
  WR2 : SIZEOF (['AIC_TOP_BND_SURF.ELEMENTARY_SURFACE',
    'AIC_TOP_BND_SURF.B_SPLINE_SURFACE',
    'AIC_TOP_BND_SURF.SWEPT_SURFACE'] *
    TYPEOF(face_geometry)) = 1;
  WR3 : SIZEOF(QUERY (elp_fbnods <* QUERY (bnds <* bounds |
    'AIC_TOP_BND_SURF.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
    NOT (SIZEOF (QUERY (oe <* elp_fbnods.bound\path.edge_list |
      NOT('AIC_TOP_BND_SURF.EDGE_CURVE' IN

```

```

        TYPEOF(oe.edge_element)))) = 0))) = 0;
WR4 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
    'AIC_TOP_BND_SURF.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
    NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
        NOT (SIZEOF ([ 'AIC_TOP_BND_SURF.LINE',
            'AIC_TOP_BND_SURF.CONIC',
            'AIC_TOP_BND_SURF.POLYLINE',
            'AIC_TOP_BND_SURF.B_SPLINE_CURVE'] *
        TYPEOF(oe.edge_element\edge_curve.edge_geometry)) = 1 )
    )) = 0));
WR5 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
    'AIC_TOP_BND_SURF.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
    NOT(SIZEOF(QUERY (oe <* elp_fbnds.bound\path.edge_list |
        NOT(( 'AIC_TOP_BND_SURF.VERTEX_POINT' IN TYPEOF(oe.edge_start)) AND
            ('AIC_TOP_BND_SURF.VERTEX_POINT' IN TYPEOF(oe.edge_end))
        ))) = 0));
WR6 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
    'AIC_TOP_BND_SURF.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
    NOT('AIC_TOP_BND_SURF.ORIENTED_PATH' IN
        TYPEOF(elp_fbnds.bound))) = 0;
WR7 : NOT (( 'AIC_TOP_BND_SURF.SWEPT_SURFACE' IN TYPEOF(face_geometry)) AND
    (NOT SIZEOF([ 'AIC_TOP_BND_SURF.LINE',
        'AIC_TOP_BND_SURF.CONIC',
        'AIC_TOP_BND_SURF.POLYLINE',
        'AIC_TOP_BND_SURF.B_SPLINE_CURVE'] *
        TYPEOF(face_geometry\swept_surface.swept_curve)) = 1));
WR8 : SIZEOF(QUERY (vlp_fbnds <* QUERY (bnds <* bounds |
    'AIC_TOP_BND_SURF.VERTEX_LOOP' IN TYPEOF(bnds.bound)) |
    NOT(( 'AIC_TOP_BND_SURF.VERTEX_POINT' IN
        TYPEOF(vlp_fbnds.bound\vertex_loop.loop_vertex)) AND
        ('AIC_TOP_BND_SURF.CARTESIAN_POINT' IN
            TYPEOF(vlp_fbnds.bound\vertex_loop.loop_vertex\vertex_point.vertex_geometry)
        ))) = 0;
WR9 : SIZEOF (QUERY (bnd <* bounds |
    NOT (SIZEOF(['AIC_TOP_BND_SURF.EDGE_LOOP',
        'AIC_TOP_BND_SURF.VERTEX_LOOP']) * TYPEOF(bnd.bound)) = 1))) = 0;
END_ENTITY;

ENTITY annotation_occurrence
  SUPERTYPE OF (annotation_text_occurrence)
  SUBTYPE OF (styled_item);
  WHERE
    wr1: SIZEOF(QUERY ( each <* USEDIN(SELF,'') | (SIZEOF(TYPEOF(each) *
        ['MECHANICAL_DESIGN_SURFACE_SCHEMA.' +
        'AREA_DEPENDENT_ANNOTATION_REPRESENTATION',
        'MECHANICAL_DESIGN_SURFACE_SCHEMA.' +

```

```

        'VIEW_DEPENDENT_ANNOTATION REPRESENTATION',
        'PRESENTATION REPRESENTATION',
        'MECHANICAL DESIGN SURFACE SCHEMA.' +
        'GEOMETRIC VIEW_DEPENDENT_ANNOTATION REPRESENTATION',
        'MECHANICAL DESIGN SURFACE SCHEMA.' +
        'CURVE_STYLE_CURVE_PATTERN',
        'MECHANICAL DESIGN SURFACE SCHEMA.' +
        'FILL_AREA_STYLE_TILE_CURVE_WITH_STYLE',
        'MECHANICAL DESIGN SURFACE SCHEMA.' +
        'FILL_AREA_STYLE_COLOURED_REGION']) = 0) )) = 0;
wr2: NOT (('MECHANICAL DESIGN SURFACE SCHEMA.' +
            'PRESENTATION_DEPENDENT_STYLED_ITEM') IN TYPEOF(SELF));
END_ENTITY; -- annotation_occurrence

ENTITY annotation_text
  SUBTYPE OF (mapped_item, geometric_representation_item);
  DERIVE
    text_source : annotation_text_map := SELF\mapped_item.mapping_source;
    placement   : axis2_placement := SELF\mapped_item.mapping_target;
END_ENTITY; -- annotation_text

ENTITY annotation_text_map
  SUBTYPE OF (representation_map);
  DERIVE
    text_string : text_string_representation := SELF\representation_map.
      mapped_representation;
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'MECHANICAL DESIGN SURFACE SCHEMA.' +
      'ANNOTATION_TEXT.TEXT_SOURCE')) > 0;
    wr2: SELF\representation_map.mapping_origin :=: SELF.text_string.
      placement;
END_ENTITY; -- annotation_text_map

ENTITY annotation_text_occurrence
  SUBTYPE OF (annotation_occurrence);
  WHERE
    wr1: 'MECHANICAL DESIGN SURFACE SCHEMA.ANNOTATION_TEXT' IN TYPEOF(
      SELF\styled_item.item);
END_ENTITY; -- annotation_text_occurrence

ENTITY application_context;
  status                      : label;
  application_interpreted_model_schema_name : label;
  application_protocol_year           : year_number;
  application                    : text;
  INVERSE

```

```

context_elements : SET [1:?] OF application_context_element FOR
    frame_of_reference;
END_ENTITY; -- application_context

ENTITY application_context_element
    SUPERTYPE OF (ONEOF (product_context,product_definition_context));
        name : label;
        frame_of_reference : application_context;
END_ENTITY; -- application_context_element

ENTITY assembly_component_usage
    SUBTYPE OF (product_definition_usage);
        reference_designator : OPTIONAL identifier;
END_ENTITY; -- assembly_component_usage

ENTITY axis1_placement
    SUBTYPE OF (placement);
        axis : OPTIONAL direction;
    DERIVE
        z : direction := NVL(normalise(axis),direction([0,0,1]));
    WHERE
        wr1: SELF\geometric_representation_item.dim = 3;
END_ENTITY; -- axis1_placement

ENTITY axis2_placement_2d
    SUBTYPE OF (placement);
        ref_direction : OPTIONAL direction;
    DERIVE
        p : LIST [2:2] OF direction := build_2axes(ref_direction);
    WHERE
        wr1: SELF\geometric_representation_item.dim = 2;
END_ENTITY; -- axis2_placement_2d

ENTITY axis2_placement_3d
    SUBTYPE OF (placement);
        axis : OPTIONAL direction;
        ref_direction : OPTIONAL direction;
    DERIVE
        p : LIST [3:3] OF direction := build_axes(axis,ref_direction);
    WHERE
        wr1: SELF\placement.location.dim = 3;
        wr2: (NOT EXISTS(axis)) OR (axis.dim = 3);
        wr3: (NOT EXISTS(ref_direction)) OR (ref_direction.dim = 3);
        wr4: ((NOT EXISTS(axis)) OR (NOT EXISTS(ref_direction))) OR (
            cross_product(axis,ref_direction).magnitude > 0);
END_ENTITY; -- axis2_placement_3d

```

```

ENTITY b_spline_curve
  SUPERTYPE OF (ONEOF (uniform_curve,quasi_uniform_curve,bezier_curve,
    b_spline_curve_with_knots) ANDOR rational_b_spline_curve)
  SUBTYPE OF (bounded_curve);
    degree          : INTEGER;
    control_points_list : LIST [2:?] OF cartesian_point;
    curve_form       : bspline_curve_form;
    closed_curve     : LOGICAL;
    self_intersect   : LOGICAL;
  DERIVE
    upper_index_on_control_points : INTEGER := SIZEOF(
      control_points_list) - 1;
    control_points : ARRAY [0:
      upper_index_on_control_points] OF
      cartesian_point := list_to_array(
      control_points_list,0,
      upper_index_on_control_points);
  WHERE
    wr1: ((('MECHANICAL_DESIGN_SURFACE_SCHEMA.UNIFORM_CURVE' IN TYPEOF(
      SELF)) OR (
      'MECHANICAL_DESIGN_SURFACE_SCHEMA.QUASI_UNIFORM_CURVE' IN
      TYPEOF(SELF))) OR (
      'MECHANICAL_DESIGN_SURFACE_SCHEMA.BEZIER_CURVE' IN TYPEOF(
      SELF))) OR ('MECHANICAL_DESIGN_SURFACE_SCHEMA.B_SPLINE_CURVE_WITH_KNOTS',
      IN TYPEOF(SELF));
  END_ENTITY; -- b_spline_curve

ENTITY b_spline_curve_with_knots
  SUBTYPE OF (b_spline_curve);
    knot_multiplicities : LIST [2:?] OF INTEGER;
    knots              : LIST [2:?] OF parameter_value;
    knot_spec          : knot_type;
  DERIVE
    upper_index_on_knots : INTEGER := SIZEOF(knots);
  WHERE
    wr1: constraints_param_bspl(degree,upper_index_on_knots,
      upper_index_on_control_points,knot_multiplicities,knots);
    wr2: SIZEOF(knot_multiplicities) = upper_index_on_knots;
  END_ENTITY; -- b_spline_curve_with_knots

ENTITY b_spline_surface
  SUPERTYPE OF (ONEOF (uniform_surface,quasi_uniform_surface,
    bezier_surface,b_spline_surface_with_knots) ANDOR
    rational_b_spline_surface)
  SUBTYPE OF (bounded_surface);

```

```

u_degree           : INTEGER;
v_degree           : INTEGER;
control_points_list : LIST [2:?] OF LIST [2:?] OF cartesian_point;
surface_form       : bspline_surface_form;
u_closed          : LOGICAL;
v_closed          : LOGICAL;
self_intersect    : LOGICAL;

DERIVE
  u_upper           : INTEGER := SIZEOF(control_points_list) - 1;
  v_upper           : INTEGER := SIZEOF(control_points_list[1]) - 1;
  control_points   : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF
    cartesian_point := make_array_of_array(
      control_points_list, 0, u_upper, 0, v_upper);

WHERE
  wr1: ((('MECHANICAL_DESIGN_SURFACE_SCHEMA.UNIFORM_SURFACE' IN
    TYPEOF(SELF)) OR (
    'MECHANICAL_DESIGN_SURFACE_SCHEMA.QUASI_UNIFORM_SURFACE' IN
    TYPEOF(SELF))) OR (
    'MECHANICAL_DESIGN_SURFACE_SCHEMA.BEZIER_SURFACE' IN TYPEOF(
    SELF))) OR ('MECHANICAL_DESIGN_SURFACE_SCHEMA.BSPLINE_SURFACE_WITH_KNOTS'
    IN TYPEOF(SELF));
END_ENTITY; -- b_spline_surface

ENTITY b_spline_surface_with_knots
  SUBTYPE OF (b_spline_surface);
  u_multiplicities : LIST [2:?] OF INTEGER;
  v_multiplicities : LIST [2:?] OF INTEGER;
  u_knots          : LIST [2:?] OF parameter_value;
  v_knots          : LIST [2:?] OF parameter_value;
  knot_spec        : knot_type;

DERIVE
  knot_u_upper : INTEGER := SIZEOF(u_knots);
  knot_v_upper : INTEGER := SIZEOF(v_knots);

WHERE
  wr1: constraints_param_bspl(SELF\b_spline_surface.u_degree,
    knot_u_upper,SELF\b_spline_surface.u_upper,u_multiplicities,
    u_knots);
  wr2: constraints_param_bspl(SELF\b_spline_surface.v_degree,
    knot_v_upper,SELF\b_spline_surface.v_upper,v_multiplicities,
    v_knots);
  wr3: SIZEOF(u_multiplicities) = knot_u_upper;
  wr4: SIZEOF(v_multiplicities) = knot_v_upper;
END_ENTITY; -- b_spline_surface_with_knots

ENTITY background_colour
  SUBTYPE OF (colour);

```

```

    presentation : area_or_view;
UNIQUE
    ur1 : presentation;
END_ENTITY; -- background_colour

ENTITY bezier_curve
    SUBTYPE OF (b_spline_curve);
END_ENTITY; -- bezier_curve

ENTITY bezier_surface
    SUBTYPE OF (b_spline_surface);
DERIVE
    ku_up : INTEGER := (u_upper / u_degree) + 1;
    kv_up : INTEGER := (v_upper / v_degree) + 1;
END_ENTITY; -- bezier_surface

ENTITY boundary_curve
    SUBTYPE OF (composite_curve_on_surface);
WHERE
    wr1: SELF\composite_curve_on_surface\composite_curve.closed_curve;
END_ENTITY; -- boundary_curve

ENTITY bounded_curve
    SUPERTYPE OF (ONEOF (polyline,b_spline_curve,trimmed_curve,
        composite_curve,composite_curve_segment))
    SUBTYPE OF (curve);
END_ENTITY; -- bounded_curve

ENTITY bounded_surface
    SUPERTYPE OF (ONEOF (b_spline_surface,rectangular_trimmed_surface,
        curve_bounded_surface,rectangular_composite_surface,surface_patch))
    SUBTYPE OF (surface);
END_ENTITY; -- bounded_surface

ENTITY camera_image
    SUBTYPE OF (geometric_representation_item,mapped_item);
DERIVE
    source   : camera_usage := SELF\mapped_item.mapping_source;
    viewport : planar_box   := SELF\mapped_item.mapping_target;
END_ENTITY; -- camera_image

ENTITY camera_model
    SUPERTYPE OF (camera_model_d3)
    SUBTYPE OF (geometric_representation_item,
        representation_item_without_style);
WHERE

```

```

wr1: SIZEOF(USEDIN(SELF,((‘MECHANICAL_DESIGN_SURFACE_SCHEMA.’ +
    ‘REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION.’) +
    ‘TRANSFORMATION_OPERATOR\ITEM_DEFINED_TRANSFORMATION.’) +
    ‘TRANSFORM_ITEM_1’) + USEDIN(SELF,
    ‘MECHANICAL_DESIGN_SURFACE_SCHEMA.’ +
    ‘CAMERA_USAGE.PROJECTION’)) > 0;
END_ENTITY; -- camera_model

ENTITY camera_model_d3
    SUBTYPE OF (camera_model);
        view_reference_system : axis2_placement_3d;
        perspective_of_volume : view_volume;
    WHERE
        wr1: (dot_product(SELF.view_reference_system.p[3],SELF.
            perspective_of_volume.view_window.placement.p[3]) = 1) AND (
            SELF.view_reference_system.location.coordinates[3] = SELF.
            perspective_of_volume.view_window.placement.location.
            coordinates[3]);
        wr2: SELF\geometric_representation_item.dim = 3;
    END_ENTITY; -- camera_model_d3

ENTITY camera_usage
    SUBTYPE OF (representation_map);
    DERIVE
        projection : camera_model := SELF\representation_map.mapping_origin;
    WHERE
        wr1: NOT (‘MECHANICAL_DESIGN_SURFACE_SCHEMA.PRESENTATION_REPRESENTATION’
            IN TYPEOF(SELF\representation_map.mapped_representation));
    END_ENTITY; -- camera_usage

ENTITY cartesian_point
    SUBTYPE OF (point);
        coordinates : LIST [1:3] OF length_measure;
    END_ENTITY; -- cartesian_point

ENTITY cartesian_transformation_operator
    SUPERTYPE OF (cartesian_transformation_operator_3d)
    SUBTYPE OF (geometric_representation_item,
        functionally_defined_transformation);
        axis1      : OPTIONAL direction;
        axis2      : OPTIONAL direction;
        local_origin : cartesian_point;
        scale       : OPTIONAL REAL;
    DERIVE
        scl : REAL := NVL(scale,1);
    WHERE

```

```

        wr1: scl > 0;
END_ENTITY; -- cartesian_transformation_operator

ENTITY cartesian_transformation_operator_3d
  SUBTYPE OF (cartesian_transformation_operator);
    axis3 : OPTIONAL direction;
  DERIVE
    u : LIST [3:3] OF direction := base_axis(3,SELF\
          cartesian_transformation_operator.axis1,SELF\
          cartesian_transformation_operator.axis2, axis3);
  WHERE
    wr1: SELF\cartesian_transformation_operator.dim = 3;
END_ENTITY; -- cartesian_transformation_operator_3d

ENTITY circle
  SUBTYPE OF (conic);
    radius : positive_length_measure;
END_ENTITY; -- circle

ENTITY closed_shell
  SUBTYPE OF (connected_face_set);
END_ENTITY; -- closed_shell

ENTITY colour;
END_ENTITY; -- colour

ENTITY colour_rgb
  SUBTYPE OF (colour_specification);
    red   : REAL;
    green : REAL;
    blue  : REAL;
  WHERE
    wr1: (0 <= red) AND (red <= 1);
    wr2: (0 <= green) AND (green <= 1);
    wr3: (0 <= blue) AND (blue <= 1);
END_ENTITY; -- colour_rgb

ENTITY colour_specification
  SUBTYPE OF (colour);
END_ENTITY; -- colour_specification

ENTITY composite_curve
  SUBTYPE OF (bounded_curve);
    segments      : LIST [1:?] OF composite_curve_segment;
    self_intersect : LOGICAL;
  DERIVE

```

```

n_segments    : INTEGER := SIZEOF(segments);
closed_curve : BOOLEAN := segments[n_segments].transition <>
                        discontinuous;
WHERE
wr1: ((NOT closed_curve) AND (SIZEOF(QUERY ( temp <* segments | (
temp.transition = discontinuous) )) = 1)) OR (closed_curve
AND (SIZEOF(QUERY ( temp <* segments | (temp.transition =
discontinuous) )) = 0));
END_ENTITY; -- composite_curve

ENTITY composite_curve_on_surface
  SUPERTYPE OF (boundary_curve)
  SUBTYPE OF (composite_curve);
  DERIVE
    basis_surface : surface := get_basis_surface(SELF\composite_curve.
                                                segments[1].parent_curve);
  WHERE
    wr1: constraints_composite_curve_on_surface(SELF);
END_ENTITY; -- composite_curve_on_surface

ENTITY composite_curve_segment
  SUBTYPE OF (bounded_curve);
  transition    : transition_code;
  same_sense    : BOOLEAN;
  parent_curve   : curve;
WHERE
wr1: (('MECHANICAL_DESIGN_SURFACE_SCHEMA.BOUNDED_CURVE' IN TYPEOF(
parent_curve)) OR (('
'MECHANICAL_DESIGN_SURFACE_SCHEMA.PCURVE' IN TYPEOF(
parent_curve)) AND (
'MECHANICAL_DESIGN_SURFACE_SCHEMA.BOUNDED_CURVE' IN TYPEOF(
parent_curve\pcurve.reference_to_curve.items[1])))) OR (('
'MECHANICAL_DESIGN_SURFACE_SCHEMA.SURFACE_CURVE' IN TYPEOF(
parent_curve)) AND (
'MECHANICAL_DESIGN_SURFACE_SCHEMA.BOUNDED_CURVE' IN TYPEOF(
parent_curve\surface_curve.curve_3d)));
wr2: NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.COMPOSITE_CURVE_SEGMENT'
IN TYPEOF(parent_curve));
END_ENTITY; -- composite_curve_segment

ENTITY conic
  ABSTRACT SUPERTYPE OF (ONEOF (circle,ellipse,hyperbola,parabola))
  SUBTYPE OF (curve);
  position : axis2_placement;
END_ENTITY; -- conic

```

```

ENTITY conical_surface
  SUBTYPE OF (elementary_surface);
    radius      : length_measure;
    semi_angle : plane_angle_measure;
  WHERE
    wr1: radius >= 0;
  END_ENTITY; -- conical_surface

ENTITY connected_face_set
  SUPERTYPE OF (ONEOF (closed_shell,open_shell))
  SUBTYPE OF (topological_representation_item);
    cfs_faces : SET [1:?] OF face;
  END_ENTITY; -- connected_face_set

ENTITY conversion_based_unit
  SUBTYPE OF (named_unit);
    name          : label;
    conversion_factor : measure_with_unit;
  END_ENTITY; -- conversion_based_unit

ENTITY curve
  SUPERTYPE OF (ONEOF (line,conic,bounded_curve,pcurve,surface_curve,
    offset_curve_3d,curve_replica))
  SUBTYPE OF (geometric_representation_item);
  END_ENTITY; -- curve

ENTITY curve_bounded_surface
  SUBTYPE OF (bounded_surface);
    basis_surface  : surface;
    boundaries     : SET [1:?] OF boundary_curve;
    implicit_outer : BOOLEAN;
  WHERE
    wr1: NOT (implicit_outer AND (
      'MECHANICAL_DESIGN_SURFACE_SCHEMA.OUTER_BOUNDARY_CURVE' IN
      TYPEOF(boundaries)));
    wr2: (NOT implicit_outer) OR (
      'MECHANICAL_DESIGN_SURFACE_SCHEMA.BOUNDED_SURFACE' IN
      TYPEOF(basis_surface));
    wr3: SIZEOF(QUERY ( temp <* boundaries | (
      'MECHANICAL_DESIGN_SURFACE_SCHEMA.OUTER_BOUNDARY_CURVE' IN
      TYPEOF(temp)) )) <= 1;
    wr4: SIZEOF(QUERY ( temp <* boundaries | (temp \
      composite_curve_on_surface.basis_surface <> SELF .
      basis_surface) )) = 0;
  END_ENTITY; -- curve_bounded_surface

```

```

ENTITY curve_replica
  SUBTYPE OF (curve);
    parent_curve : curve;
    transformation : cartesian_transformation_operator;
  WHERE
    wr1: transformation.dim = parent_curve.dim;
  END_ENTITY; -- curve_replica

ENTITY curve_style;
  curve_font : curve_font_or_scaled_curve_font_select;
  curve_width : size_select;
  curve_colour : colour;
END_ENTITY; -- curve_style

ENTITY curve_style_font;
  pattern_list : LIST [1:?] OF curve_style_font_pattern;
END_ENTITY; -- curve_style_font

ENTITY curve_style_font_pattern;
  visible_segment_length : positive_length_measure;
  invisible_segment_length : positive_length_measure;
END_ENTITY; -- curve_style_font_pattern

ENTITY curve_style_rendering;
  rendering_method : shading_curve_method;
  rendering_properties : surface_rendering_properties;
END_ENTITY; -- curve_style_rendering

ENTITY cylindrical_surface
  SUBTYPE OF (elementary_surface);
  radius : positive_length_measure;
END_ENTITY; -- cylindrical_surface

ENTITY definitional_representation_item;
  items : SET [1:?] OF representation_item;
  context_of_items : parametric_representation_context;
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'')) > 0;
  END_ENTITY; -- definitional_representation_item

ENTITY degenerate_pcurve
  SUBTYPE OF (point);
  basis_surface : surface;
  reference_to_curve : definitional_representation_item;
  WHERE
    wr1: SIZEOF(reference_to_curve.items) = 1;

```

```

wr2: 'MECHANICAL_DESIGN_SURFACE_SCHEMA.CURVE' IN TYPEOF(
    reference_to_curve.items[1]);
wr3: reference_to_curve.items[1]\geometric_representation_item.dim =
    2;
END_ENTITY; -- degenerate_pcurve

ENTITY design_context
    SUBTYPE OF (product_definition_context);
        aic_functionality : LIST [2:2] OF STRING;
    WHERE
        wr1: aic_functionality =
            aic_mechanical_design_context_functionality_definition;
        wr2: SELF.life_cycle_stage = 'design';
    END_ENTITY; -- design_context

ENTITY dimensional_exponents;
    length_exponent           : REAL;
    mass_exponent              : REAL;
    time_exponent              : REAL;
    electric_current_exponent : REAL;
    thermodynamic_temperature_exponent : REAL;
    amount_of_substance_exponent : REAL;
    luminous_intensity_exponent : REAL;
END_ENTITY; -- dimensional_exponents

ENTITY direction
    SUBTYPE OF (geometric_representation_item);
        direction_ratios : LIST [2:3] OF REAL;
    WHERE
        wr1: NOT (((direction_ratios[1] = 0) AND (direction_ratios[2] = 0))
                   AND (direction_ratios[3] = 0));
    END_ENTITY; -- direction

ENTITY edge
    SUPERTYPE OF (ONEOF (edge_curve, oriented_edge))
    SUBTYPE OF (topological_representation_item);
        edge_start : vertex;
        edge_end   : vertex;
    END_ENTITY; -- edge

ENTITY edge_curve
    SUBTYPE OF (edge, geometric_representation_item);
        edge_geometry : curve;
        same_sense   : BOOLEAN;
    END_ENTITY; -- edge_curve

```

```

ENTITY edge_loop
  SUBTYPE OF (loop, path);
  DERIVE
    ne : INTEGER := SIZEOF(SELF\path.edge_list);
  WHERE
    wr1: SELF\path.edge_list[1].edge_element.edge_start ::= SELF\path.
      edge_list[ne].edge_element.edge_end;
  END_ENTITY; -- edge_loop

ENTITY elementary_surface
  SUPERTYPE OF (ONEOF (plane,cylindrical_surface,conical_surface,
    spherical_surface,toroidal_surface))
  SUBTYPE OF (surface);
  position : axis2_placement_3d;
END_ENTITY; -- elementary_surface

ENTITY ellipse
  SUBTYPE OF (conic);
  semi_axis_1 : positive_length_measure;
  semi_axis_2 : positive_length_measure;
END_ENTITY; -- ellipse

ENTITY evaluated_degenerate_pcurve
  SUBTYPE OF (degenerate_pcurve);
  equivalent_point : cartesian_point;
END_ENTITY; -- evaluated_degenerate_pcurve

ENTITY face
  SUPERTYPE OF (ONEOF (face_surface,oriented_face))
  SUBTYPE OF (topological_representation_item);
  bounds : SET [1:?] OF face_bound;
  WHERE
    wr1: NOT mixed_loop_type_set(list_to_set(list_face_loops(SELF)));
    wr2: SIZEOF(QUERY ( temp <* bounds | (
      'MECHANICAL_DESIGN_SURFACE_SCHEMA.FACE_OUTER_BOUND' IN
      TYPEOF(temp)) )) <= 1;
  END_ENTITY; -- face

ENTITY face_based_surface_model
  SUBTYPE OF (geometric_representation_item);
  fbsm_faces : SET [1:?] OF connected_face_set;
END_ENTITY; -- face_based_surface_model

ENTITY face_bound
  SUBTYPE OF (topological_representation_item);
  bound       : loop;

```

```

        orientation : BOOLEAN;
END_ENTITY; -- face_bound

ENTITY face_outer_bound
  SUBTYPE OF (face_bound);
END_ENTITY; -- face_outer_bound

ENTITY face_surface
  SUBTYPE OF (face, geometric_representation_item);
  face_geometry : surface;
  same_sense    : BOOLEAN;
END_ENTITY; -- face_surface

ENTITY functionally_defined_transformation;
END_ENTITY; -- functionally_defined_transformation

ENTITY geometric_representation_context
  SUBTYPE OF (representation_context);
  coordinate_space_dimension : dimension_count;
END_ENTITY; -- geometric_representation_context

ENTITY geometric_representation_item
  SUBTYPE OF (representation_item);
  DERIVE
    dim : dimension_count := dimension_of(SELF);
END_ENTITY; -- geometric_representation_item

ENTITY geometric_set
  SUPERTYPE OF (geometric_set_replica)
  SUBTYPE OF (geometric_representation_item);
  elements : SET [1:?] OF geometric_set_select;
END_ENTITY; -- geometric_set

ENTITY geometric_set_replica
  SUBTYPE OF (geometric_set);
  parent_set      : geometric_set;
  transformation : cartesian_transformation_operator;
  DERIVE
    SELF\geometric_set.elements : SET [1:?] OF geometric_set_select :=
      build_transformed_set(transformation,
      parent_set);
END_ENTITY; -- geometric_set_replica

ENTITY geometrically_bounded_surface_shape_representation
  SUBTYPE OF (shape_representation);
  aic_functionality : LIST [2:2] OF STRING;

```

WHERE

```

WR1 : aic_functionality =
      aic_geometrically_bounded_functionality_definition;

WR2 : -- legal_items FOR
      -- (geometrically_bounded_surface_shape_representation);
      SIZEOF (QUERY (it <* items |
      NOT (SIZEOF (['AIC_GBND_SURF.GEOMETRIC_SET',
      'AIC_GBND_SURF.MAPPED_ITEM',
      'AIC_GBND_SURF.AXIS2_PLACEMENT_3D'] * TYPEOF (it)) = 1))) = 0;

WR3 : -- minimum_requirement FOR
      -- (geometrically_bounded_surface_shape_representation);
      SIZEOF (QUERY (it <* items |
      SIZEOF (['AIC_GBND_SURF.GEOMETRIC_SET',
      'AIC_GBND_SURF.MAPPED_ITEM'] * TYPEOF (it)) = 1)) > 0;

WR4 : -- legal_mapped_representation FOR (mapped_item);
      SIZEOF (QUERY (mi <* QUERY (it <* items |
      'AIC_GBND_SURF.MAPPED_ITEM' IN TYPEOF (it)) |
      NOT ('AIC_GBND_SURF.' +
      'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION'
      IN TYPEOF (mi\mapped_item.mapping_source.
      mapped_representation))) = 0;

WR5 : -- legal_point FOR (point)
      SIZEOF (QUERY (gs <* QUERY (it <* items |
      'AIC_GBND_SURF.GEOMETRIC_SET' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (pnt <* QUERY (gsel <* gs.elements |
      'AIC_GBND_SURF.POINT' IN TYPEOF (gsel)) |
      NOT (gbsf_check_point(pnt, 'AIC_GBND_SURF')))) = 0))) = 0;

WR6 : -- legal_curve FOR (curve)
      SIZEOF (QUERY (gs <* QUERY (it <* items |
      'AIC_GBND_SURF.GEOMETRIC_SET' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cv <* QUERY (gsel <* gs.elements |
      'AIC_GBND_SURF.CURVE' IN TYPEOF (gsel)) |
      NOT (gbsf_check_curve(cv, 'AIC_GBND_SURF')))) = 0))) = 0;

WR7 : -- legal_surface FOR (surface)
      SIZEOF (QUERY (gs <* QUERY (it <* items |
      'AIC_GBND_SURF.GEOMETRIC_SET' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (sf <* QUERY (gsel <* gs.elements |
      'AIC_GBND_SURF.SURFACE' IN TYPEOF (gsel)) |
      NOT (gbsf_check_surface(sf, 'AIC_GBND_SURF')))) = 0))) = 0;

```

```

END_ENTITY; -- geometrically_boundedsurface_shape_representation

ENTITY global_unit_assigned_context
  SUBTYPE OF (representation_context);
    units : SET [1:?] OF unit;
END_ENTITY; -- global_unit_assigned_context

ENTITY group;
  name : label;
END_ENTITY; -- group

ENTITY group_assignment
  ABSTRACT SUPERTYPE;
  assigned_group : group;
END_ENTITY; -- group_assignment

ENTITY hyperbola
  SUBTYPE OF (conic);
  semi_axis      : positive_length_measure;
  semi_imag_axis : positive_length_measure;
END_ENTITY; -- hyperbola

ENTITY intersection_curve
  SUBTYPE OF (surface_curve);
  WHERE
    wr1: SIZEOF(SELF\surface_curve.associated_geometry) = 2;
    wr2: associated_surface(associated_geometry[1]) <>
          associated_surface(associated_geometry[2]);
END_ENTITY; -- intersection_curve

ENTITY length_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: (((((SELF.dimensions.length_exponent = 1) AND (SELF.dimensions
      .mass_exponent = 0)) AND (SELF.dimensions.time_exponent = 0))
      AND (SELF.dimensions.electric_current_exponent = 0)) AND (
      SELF.dimensions.thermodynamic_temperature_exponent = 0)) AND
      (SELF.dimensions.amount_of_substance_exponent = 0)) AND (
      SELF.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- length_unit

ENTITY line
  SUBTYPE OF (curve);
  pnt : cartesian_point;
  dir : vector;
  WHERE

```

```

wr1: dir.dim = pnt.dim;
END_ENTITY; -- line

ENTITY loop
  SUPERTYPE OF (ONEOF (vertex_loop,edge_loop))
  SUBTYPE OF (topological_representation_item);
END_ENTITY; -- loop

ENTITY manifold_surface_shape_representation
  SUBTYPE OF (shape_representation);
  aic_functionality : LIST [2:2] OF STRING;
WHERE
WR1 : aic_functionality =
      aic_manifold_surface_functionality_definition;

WR2 : -- legal_items FOR (manifold_surface_shape_representation);
      SIZEOF (QUERY (it <* items |
      NOT (SIZEOF (['AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL',
      'AIC_MFLD_SURF.MAPPED_ITEM',
      'AIC_MFLD_SURF.AXIS2_PLACEMENT_3D'] * TYPEOF (it)) = 1))) = 0;

WR3 : -- minimum_requirement FOR (manifold_surface_shape_representation);
      SIZEOF (QUERY (it <* items |
      SIZEOF (['AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL',
      'AIC_MFLD_SURF.MAPPED_ITEM'] * TYPEOF (it)) = 1)) > 0;

WR4 : -- legal_mapped_representation FOR (mapped_item);
      SIZEOF (QUERY (mi <* QUERY (it <* items |
      'AIC_MFLD_SURF.MAPPED_ITEM' IN TYPEOF (it)) |
      NOT
      ('AIC_MFLD_SURF.MANIFOLD_SURFACE_SHAPE_REPRESENTATION'
      IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation)))
      = 0;

WR5 : -- legal_selects_for_shell FOR (shell_based_surface_model);
      SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (sh <* sbsm.sbsm_boundary |
      NOT (SIZEOF (['AIC_MFLD_SURF.OPEN_SHELL','AIC_MFLD_SURF.CLOSED_SHELL']
      * TYPEOF (sh)) = 1))) = 0))) = 0;

WR6 : -- legal_subtypes_of_face FOR (connected_face_set);
      SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
      'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs)) |

```

```

        NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
        NOT (SIZEOF (['AIC_MFLD_SURF.FACE_SURFACE',
        'AIC_MFLD_SURF.ORIENTED_FACE'] * TYPEOF (fa)) = 1))) = 0)))
= 0))) = 0;

WR7 : -- legal_face_geometry FOR (face_surface);
      SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
      'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs)) |
      NOT (SIZEOF (QUERY (f_sf <* QUERY (fa <* cfs.cfs_faces |
      'AIC_MFLD_SURF.FACE_SURFACE' IN TYPEOF (fa)) |
      NOT ('AIC_MFLD_SURFADVANCED_FACE' IN TYPEOF (f_sf))
      OR
      (SIZEOF (['AIC_MFLD_SURF.OFFSET_SURFACE',
      'AIC_MFLD_SURF.SURFACE_REPLICA'] * TYPEOF
      (f_sf.face_geometry)) = 1)))) = 0))) = 0))) = 0;

WR8 : -- legal_basis_surface FOR (all surfaces that are
      -- referenced from face);
      SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
      'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs)) |
      NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
      NOT ('AIC_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
      OR
      basis_surface_check(fa.face_geometry,'AIC_MFLD_SURF'))))
= 0))) = 0);

WR9 : -- legal_loop_type FOR (loop);
      SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
      NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
      'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs)) |
      NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
      NOT ('AIC_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
      OR
      (SIZEOF (QUERY (bnds <* fa.bounds |
      NOT (SIZEOF (['AIC_MFLD_SURF.EDGE_LOOP',
      'AIC_MFLD_SURF.VERTEX_LOOP']
      * TYPEOF (bnds.bound)) = 1)))) = 0)))) = 0))) = 0;

WR10: -- mandatory_edge_geometry FOR (oriented_edge);
      SIZEOF (QUERY (sbsm <* QUERY (it <* items |
      'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |

```

```

NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs) |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT ('AIC_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds)) |
NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
NOT ('AIC_MFLD_SURF.EDGE_CURVE' IN TYPEOF (oe.edge_element)))
= 0))) = 0))) = 0))) = 0;

WR11: -- legal_edge_geometry FOR (edge_curve);
SIZEOF (QUERY (sbsm <* QUERY (it <* items |
'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs) |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT ('AIC_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
NOT (SIZEOF (QUERY (e_cv <* QUERY (oe <*
elp_fbnds.bound\path.edge_list |
'AIC_MFLD_SURF.EDGE_CURVE' IN TYPEOF (oe.edge_element)) |
NOT (SIZEOF (['AIC_MFLD_SURF.CURVE_REPLICA',
'AIC_MFLD_SURF.OFFSET_CURVE_3D', 'AIC_MFLD_SURF.PCURVE',
'AIC_MFLD_SURF.SURFACE_CURVE'] * TYPEOF (e_cv.edge_geometry))
= 1))) = 0))) = 0))) = 0))) = 0;

WR12: -- legal_basis_curve FOR (all curves that are
-- referenced from edge);
SIZEOF (QUERY (sbsm <* QUERY (it <* items |
'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs) |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
NOT (basis_curve_check(oe.edge_element.edge_geometry,
'AIC_MFLD_SURF')))) = 0))) = 0))) = 0))) = 0;

WR13: -- legal_basis_surface FOR (pcurve and surface_curve);
SIZEOF (QUERY (sbsm <* QUERY (it <* items |
'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |

```

```

'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs) |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
NOT (SIZEOF (QUERY (p_oe_geom <* QUERY (oe <*
elp_fbnds.bound\path.edge_list | SIZEOF (['AIC_MFLD_SURF.PCURVE',
'AIC_MFLD_SURF.SURFACE_CURVE'] *
TYPEOF (oe.edge_element.edge_geometry)) = 1 |
NOT (SIZEOF (['AIC_MFLD_SURF.B_SPLINE_SURFACE',
'AIC_MFLD_SURF.ELEMENTARY_SURFACE',
'AIC_MFLD_SURF.OFFSET_SURFACE',
'AIC_MFLD_SURF.SURFACE_REPLICA',
'AIC_MFLD_SURF.SWEPT_SURFACE'] *
* TYPEOF (p_oe_geom.basis_surface)) = 1))) = 0 ))) =
= 0 )))) = 0 )))) = 0;

WR14: -- legal_basis_curve_for_swept_surface FOR (swept_surface
--);
SIZEOF (QUERY (sbsm <* QUERY (it <* items |
'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs) |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
'AIC_MFLD_SURF.FACE_SURFACE' IN TYPEOF (fa)) |
NOT (('AIC_MFLD_SURFADVANCED_FACE' IN TYPEOF (f_sf))
OR
(SIZEOF (QUERY (sw_sf <* f_sf.face_geometry |
'AIC_MFLD_SURF.SWEPT_SURFACE' IN TYPEOF (sw_sf)) |
NOT (SIZEOF (['AIC_MFLD_SURF.B_SPLINE_CURVE',
'AIC_MFLD_SURF.CONIC', 'AIC_MFLD_SURF.CURVE_REPLICA',
'AIC_MFLD_SURF.LINE', 'AIC_MFLD_SURF.OFFSET_CURVE_3D',
'AIC_MFLD_SURF.PCURVE', 'AIC_MFLD_SURF.POLYLINE',
'AIC_MFLD_SURF.SURFACE_CURVE'] * TYPEOF (sw_sf.swept_curve))
= 1)))) = 0)))) = 0))) = 0;

WR15: -- mandatory_vertex_geometry_in_edge_loop FOR (vertex);
SIZEOF (QUERY (sbsm <* QUERY (it <* items |
'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs) |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT (('AIC_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |

```

```

NOT ('AIC_MFLD_SURF.VERTEX_POINT' IN TYPEOF
(oe.edge_element.edge_start)
AND
'AIC_MFLD_SURF.VERTEX_POINT' IN TYPEOF (oe.edge_element.edge_end)))
= 0))) = 0))) = 0))) = 0))) = 0;

WR16: -- legal_vertex_geometry_in_edge_loop FOR (vertex_point);
SIZEOF (QUERY (sbsm <* QUERY (it <* items |
'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs)) |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT ('AIC_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
NOT (SIZEOF ([ 'AIC_MFLD_SURF.CARTESIAN_POINT',
'AIC_MFLD_SURF.DEGENERATE_PCURVE',
'AIC_MFLD_SURF.POINT_ON_CURVE',
'AIC_MFLD_SURF.POINT_ON_SURFACE'] * TYPEOF
(oe.edge_element.edge_start)) = 1
AND
(SIZEOF ([ 'AIC_MFLD_SURF.CARTESIAN_POINT',
'AIC_MFLD_SURF.DEGENERATE_PCURVE',
'AIC_MFLD_SURF.POINT_ON_CURVE',
'AIC_MFLD_SURF.POINT_ON_SURFACE'] * TYPEOF
(oe.edge_element.edge_end))
= 1)))) = 0))) = 0))) = 0))) = 0;

WR17: -- mandatory_vertex_geometry_in_vertex_loop FOR (vertex);
SIZEOF (QUERY (sbsm <* QUERY (it <* items |
'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs)) |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT ('AIC_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_MFLD_SURF.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
NOT ('AIC_MFLD_SURF.VERTEX_POINT' IN TYPEOF
(vlp_fbnds.loop_vertex)))) = 0)))) = 0))) = 0;

WR18: -- legal_vertex_geometry_in_vertex_loop FOR (vertex_point);
SIZEOF (QUERY (sbsm <* QUERY (it <* items |
'AIC_MFLD_SURF.SHELL_BASED_SURFACE_MODEL' IN TYPEOF (it)) |

```

```

        NOT (SIZEOF (QUERY (cfs <* sbsm.sbsm_boundary |
        'AIC_MFLD_SURF.CONNECTED_FACE_SET' IN TYPEOF (cfs) |
        NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
        NOT (('AIC_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
        OR
        (SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* fa.bounds |
        'AIC_MFLD_SURF.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
        NOT (SIZEOF (['AIC_MFLD_SURFCARTESIAN_POINT',
        'AIC_MFLD_SURF.DEGENERATE_PCURVE',
        'AIC_MFLD_SURF.POINT_ON_CURVE',
        'AIC_MFLD_SURF.POINT_ON_SURFACE'] * TYPEOF
        (vlp_fbnds.loop_vertex)) = 1))) = 0)))) = 0))) = 0));
END_ENTITY; -- manifold_surface_shape_representation

ENTITY mapped_item
  SUBTYPE OF (representation_item);
  mapping_source : representation_map;
  mapping_target : representation_item;
WHERE
  wr1: acyclic_mapped_representation(using_representations(SELF),[SELF]);
END_ENTITY; -- mapped_item

ENTITY measure_with_unit;
  value_component : measure_value;
  unit_component : unit;
WHERE
  wr1: valid_units(SELF);
END_ENTITY; -- measure_with_unit

ENTITY mechanical_context
  SUBTYPE OF (product_context);
  aic_functionality : LIST [2:2] OF STRING;
WHERE
  wr1: aic_functionality =
    aic_mechanical_design_context_functionality_definition;
  wr2: SELF.discipline_type = 'mechanical';
END_ENTITY; -- mechanical_context

ENTITY mechanical_design_group_assignment
  SUBTYPE OF (group_assignment);
  items : SET [1:?] OF grouped_item_select;
END_ENTITY; -- mechanical_design_group_assignment

ENTITY mechanical_design_name_assignment
  SUBTYPE OF (name_assignment);
  items : SET [1:?] OF named_item_select;

```

```

END_ENTITY; -- mechanical_design_name_assignment

ENTITY mechanical_design_pre_defined_text_font
  SUBTYPE OF (pre_defined_text_font);
WHERE
  WR1: SELF.name IN ['ISO 3098 font'];
END_ENTITY; -- mechanical_design_pre_defined_text_font

ENTITY mechanical_design_pre_defined_text_style
  SUBTYPE OF (pre_defined_presentation_style);
WHERE
  WR1 : SELF.name IN ['mechanical design text style'];
END_ENTITY; -- mechanical_design_pre_defined_text_style

ENTITY mechanical_design_presentation_area
  SUBTYPE OF (presentation_area);
  aic_functionality : LIST [2:2] OF STRING;
WHERE
  WR1 : aic_functionality =
    aic_mechanical_design_presentation_functionality_definition;

WR2 : -- legal_representations FOR (camera_usage);
-- get for all presentation_areas their presentation_views
SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* items |
  'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it1)) |
  'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW' IN TYPEOF
  (mi1\mapped_item.mapping_source.mapped_representation)) |
-- get for all presentation_views their
-- product_data_representation_views
NOT (SIZEOF (QUERY (pdrv <* QUERY (mi2 <* QUERY (it2 <*
  pv.items |
  'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it2)) |
  'AIC_MECH_DSGN_PRES.PRODUCT_DATA_REPRESENTATION_VIEW' IN TYPEOF
  (mi2\mapped_item.mapping_source.mapped_representation)) |
-- a product_data_representation_view has exactly one item in its list
-- of items and this is a camera_image;
-- get the camera_usage of this camera_image
NOT (SIZEOF (QUERY (cu <* QUERY (it3 <* pdrv.items |
  'AIC_MECH_DSGN_PRES.CAMERA_USAGE' IN TYPEOF
  (it3\mapped_item.mapping_source)) |
-- apply the test that a camera_usage shall have a
-- shape_representation or a
-- mechanical_design_presentation_representation as its
-- mapped_representation
NOT (SIZEOF (['AIC_MECH_DSGN_PRES.SHAPE_REPRESENTATION',
  'AIC_MECH_DSGN_PRES.MECHANICAL DESIGN_PRESENTATION_REPRESENTATION'])

```

```

* TYPEOF (cu\representation_map.mapped_representation)) = 1 )))  

= 0 ))) = 0 )) = 0;

WR3 : -- mandatory_curve_style FOR (curved_styled_item);  

-- get for all presentation_areas their presentation_views  

SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* items |  

'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it1)) |  

'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW' IN TYPEOF  

(mi1\mapped_item.mapping_source.mapped_representation)) |  

-- get for all presentation_views their  

-- product_data_representation_views  

NOT (SIZEOF (QUERY (pdrv <* QUERY (mi2 <* QUERY (it2 <*  

pv.items |  

'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it2)) |  

'AIC_MECH_DSGN_PRES.PRODUCT_DATA REPRESENTATION_VIEW' IN TYPEOF  

(mi2\mapped_item.mapping_source.mapped_representation)) |  

-- a product_data_representation_view has exactly one item in its list  

-- of items and this is a camera_image; get the camera_usage of this  

-- camera_image and the mapped_representation of type  

-- shape_representation  

NOT (SIZEOF (QUERY (sr <* QUERY (cu <* QUERY (it3 <* pdrv.items |  

'AIC_MECH_DSGN_PRES.CAMERA_USAGE' IN TYPEOF  

(it3\mapped_item.mapping_source)) |  

'AIC_MECH_DSGN_PRES.SHAPE REPRESENTATION'  

IN TYPEOF (cu\representation_map.mapped_representation)) |  

-- get all items of the mapped_representation of the camera_usage;  

-- filter out those of type styled_items and out of these the curves  

NOT (SIZEOF (QUERY (cv <* QUERY (si <* QUERY (it4 <* sr.items |  

'AIC_MECH_DSGN_PRES.STYLED_ITEM' IN TYPEOF (it4)) |  

'GEOMETRY_SCHEMA.CURVE' IN TYPEOF (si.item)) |  

-- apply the restriction that a curve in styled_item needs a  

-- curve_style  

NOT (SIZEOF (QUERY (psa <* si.styles | SIZEOF (psa.styles) = 1 AND  

'AIC_MECH_DSGN_PRES.CURVE_STYLE' IN TYPEOF (psa.styles[1])))  

= 0 ))) = 0 ))) = 0 )) = 0)) = 0;

WR4 : -- mandatory_point_style FOR (pointed_styled_item);  

-- get for all presentation_areas their presentation_views  

SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* items |  

'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it1)) |  

'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW' IN TYPEOF  

(mi1\mapped_item.mapping_source.mapped_representation)) |  

-- get for all presentation_views their  

-- product_data_representation_views  

NOT (SIZEOF (QUERY (pdrv <* QUERY (mi2 <* QUERY (it2 <*  

pv.items |

```

```

'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it2)) |
'AIC_MECH_DSGN_PRES.PRODUCT_DATA REPRESENTATION_VIEW' IN TYPEOF
(mi2\mapped_item.mapping_source.mapped_representation)) |
-- a product_data_representation_view has exactly one item in its list
-- of items and this is a camera_image; get the camera_usage of this
-- camera_image and the mapped_representation of type
-- shape_representation
NOT (SIZEOF (QUERY (sr <* QUERY (cu <* QUERY (it3 <* pdrv.items |
'AIC_MECH_DSGN_PRES.CAMERA_USAGE' IN TYPEOF
(it3\mapped_item.mapping_source)) |
'AIC_MECH_DSGN_PRES.SHAPE_REPRESENTATION'
IN TYPEOF (cu\representation_map.mapped_representation)) |
-- get all items of the mapped_representation of the camera_usage;
-- filter out those of type styled_items and out of these the points
NOT (SIZEOF (QUERY (pnt <* QUERY (si <* QUERY (it4 <* sr.items |
'AIC_MECH_DSGN_PRES.STYLED_ITEM' IN TYPEOF (it4)) |
'GEOMETRY_SCHEMA.POINT' IN TYPEOF (si.item)) |
-- apply the restriction that a point in styled_item needs a
-- point_style
NOT (SIZEOF (QUERY (psa <* si.styles | SIZEOF (psa.styles) = 1 AND
'AIC_MECH_DSGN_PRES.POINT_STYLE' IN TYPEOF (psa.styles[1])))
= 0 ))) = 0 )))) = 0 )))) = 0;

WR5 : -- predefined_text_style FOR (styled_annotation_text);
-- get for all presentation_areas their presentation_views
SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* items |
'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it1)) |
'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW' IN TYPEOF
(mi1\mapped_item.mapping_source.mapped_representation)) |
-- get for all presentation_views their
-- view_dependent_annotation_representations
NOT (SIZEOF (QUERY (vdar <* QUERY (mi2 <* QUERY (it2 <*
pv.items |
'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it2)) |
'AIC_MECH_DSGN_PRES.VIEW_DEPENDENT_ANNOTATION_REPRESENTATION'
IN TYPEOF
(mi2\mapped_item.mapping_source.mapped_representation)) |
-- get for all view_dependent_annotation_representations their
-- annotation_text_occurrences
NOT (SIZEOF (QUERY (atc <* QUERY (it3 <* vdar.items |
'AIC_MECH_DSGN_PRES.ANNOTATION_TEXT_OCCURRENCE' IN TYPEOF (it3)) |
-- apply the test that all annotation_text_occurrence shall reference
-- an annotation_text and shall have only one style that is the
-- mechanical_design_pre_defined_text_style
NOT ('AIC_MECH_DSGN_PRES.ANNOTATION_TEXT' IN TYPEOF (atc.item)) AND
NOT (SIZEOF (QUERY (psa <* atc.styles |

```

```

        NOT (SIZEOF (psa.styles) = 1) AND
        NOT ('AIC_MECH_DSGN_PRES.MECHANICAL DESIGN_PRE_DEFINED_TEXT_STYLE'
        IN TYPEOF (psa.styles[1])))) = 0 ))) = 0 )))) = 0;

WR6 : -- mandatory_mech_des_pre_text_font FOR (annotation_text);
-- get for all presentation_areas their presentation_views
SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* items |
'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it1)) |
'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW' IN TYPEOF
(mi1\mapped_item.mapping_source.mapped_representation)) |
-- get for all presentation_views their
-- view_dependent_annotation_representations
NOT (SIZEOF (QUERY (vdar <* QUERY (mi2 <* QUERY (it2 <*
pv.items |
'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it2)) |
'AIC_MECH_DSGN_PRES.VIEW_DEPENDENT_ANNOTATION_REPRESENTATION'
IN TYPEOF
(mi2\mapped_item.mapping_source.mapped_representation)) |
-- get for all view_dependent_annotation_representations their
-- annotation_text_occurrences
NOT (SIZEOF (QUERY (atc <* QUERY (it3 <* vdar.items |
'AIC_MECH_DSGN_PRES.ANNOTATION_TEXT_OCCURRENCE' IN TYPEOF (it3)) |
-- apply the restriction that each annotation_text shall have the
-- mechanical_design_pre_defined_text_font; that the item of an
-- annotation_text_occurrence is an annotation_text has been checked
-- above; use function to cover the recursion that an annotation_text
-- may be in the list of strings of a text_literal_mapped_item
NOT (SIZEOF (QUERY (tlmi <*
mdp_get_text_literal_mapped_item_in_annotation_text
(atc.item, 'AIC_MECH_DSGN_PRES') |
'AIC_MECH_DSGN_PRES.MECHANICAL DESIGN_PRE_DEFINED_TEXT_FONT'
IN TYPEOF (tlmi\mapped_item.mapped_representation.font))) =
0 ))) = 0 ))) = 0 )))) = 0;

END_ENTITY; -- mechanical_design_presentation_area

ENTITY mechanical_design_presentation_representation
  SUBTYPE OF (presentation_representation);
WHERE
  WR1 : SIZEOF (QUERY (it <* SELF.items |
    NOT ('AIC_MECH_DSGN_PRES.STYLED_ITEM' IN TYPEOF (it)))) = 0;
END_ENTITY; -- mechanical_design_presentation_representation

ENTITY name_assignment
  ABSTRACT SUPERTYPE;
  assigned_name : label;

```

```

END_ENTITY; -- name_assignment

ENTITY named_unit
  SUPERTYPE OF (ONEOF (si_unit,conversion_based_unit));
  dimensions : dimensional_exponents;
END_ENTITY; -- named_unit

ENTITY non_manifold_surface_shape_representation
  SUBTYPE OF (shape_representation);
  aic_functionality : LIST [2:2] OF STRING;
WHERE
  WR1 : aic_functionality =
    aic_non_manifold_surface_functionality_definition;

  WR2 : -- legal_items FOR (non_manifold_surface_shape_representation);
  SIZEOF (QUERY (it <* items |
  NOT (SIZEOF (['AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL',
  'AIC_NON_MFLD_SURF.MAPPED_ITEM',
  'AIC_NON_MFLD_SURF.AXIS2_PLACEMENT_3D'] * TYPEOF (it)) = 1))) = 0;

  WR3 : -- minimum_requirement FOR (non_manifold_surface_shape_representation);
  SIZEOF (QUERY (it <* items |
  SIZEOF (['AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL',
  'AIC_NON_MFLD_SURF.MAPPED_ITEM'] * TYPEOF (it)) = 1)) > 0;

  WR4 : -- legal_mapped_representation FOR (mapped_item);
  SIZEOF (QUERY (mi <* QUERY (it <* items |
  'AIC_NON_MFLD_SURF.MAPPED_ITEM' IN TYPEOF (it)) |
  NOT ('AIC_NON_MFLD_SURF.NON_MANIFOLD_SURFACE_SHAPE_REPRESENTATION' IN
  TYPEOF (mi\mapped_item.mapping_source.mapped_representation))) = 0;

  WR5 : -- legal_subtypes_of_face FOR (connected_face_set);
  SIZEOF (QUERY (fbsm <* QUERY (it <* items |
  'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
  NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
  NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
  NOT (SIZEOF (['AIC_NON_MFLD_SURF.FACE_SURFACE',
  'AIC_NON_MFLD_SURF.ORIENTED_FACE'] * TYPEOF (fa)) = 1))) =
  0)))) = 0)) = 0;

  WR6 : -- legal_face_geometry FOR (face_surface);
  SIZEOF (QUERY (fbsm <* QUERY (it <* items |
  'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
  NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
  NOT (SIZEOF (QUERY (f_sf <* QUERY (fa <* cfs.cfs_faces |
  'AIC_NON_MFLD_SURF.FACE_SURFACE' IN TYPEOF (fa)) |
```

```

NOT (('AIC_NON_MFLD_SURF.ADVANCED_FACE' IN TYPEOF (f_sf))
OR
(SIZEOF ([ 'AIC_NON_MFLD_SURF.OFFSET_SURFACE',
'AIC_NON_MFLD_SURF.SURFACE_REPLICA'] * TYPEOF
(f_sf.face_geometry)) = 1))) = 0))) = 0;

WR7: -- legal_basis_surface FOR (all surfaces that are referenced from face);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT (('AIC_NON_MFLD_SURF.ADVANCED_FACE' IN TYPEOF (fa))
OR
basis_surface_check(fa.face_geometry,'AIC_NON_MFLD_SURF'))))
= 0))) = 0)) = 0;

WR8 : -- legal_loop_type FOR (loop);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT (('AIC_NON_MFLD_SURF.ADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (bnds <* fa.bounds |
NOT (SIZEOF ([ 'AIC_NON_MFLD_SURF.EDGE_LOOP',
'AIC_NON_MFLD_SURF.VERTEX_LOOP']
* TYPEOF (bnds.bound)) = 1)))) = 0)))) = 0))) = 0;

WR9 : -- mandatory_edge_geometry FOR (oriented_edge);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT (('AIC_NON_MFLD_SURF.ADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (elp_fbnd <* QUERY (bnds <* fa.bounds |
'AIC_NON_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds)) |
NOT (SIZEOF (QUERY (oe <* elp_fbnd.bound\path.edge_list |
NOT ('AIC_NON_MFLD_SURF.EDGE_CURVE' IN TYPEOF (oe.edge_element)))) =
0)))) = 0)))) = 0))) = 0);

WR10: -- legal_edge_geometry FOR (edge_curve);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |

```

```

NOT (('AIC_NON_MFLD_SURF.ADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_NON_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
NOT (SIZEOF (QUERY (e_cv <* QUERY (oe <*
elp_fbnds.bound\path.edge_list |
'AIC_NON_MFLD_SURF.EDGE_CURVE' IN TYPEOF (oe.edge_element)) |
NOT (SIZEOF (['AIC_NON_MFLD_SURF.CURVE_REPLICA',
'AIC_NON_MFLD_SURF.OFFSET_CURVE_3D', 'AIC_NON_MFLD_SURF.PCURVE',
'AIC_NON_MFLD_SURF.SURFACE_CURVE'] * TYPEOF (e_cv.edge_geometry))
= 1))) = 0)))) = 0))) = 0;

WR11: -- legal_basis_curve FOR (all curves that are referenced from edge);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_NON_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
NOT (basis_curve_check(oe.edge_element.edge_geometry,
'AIC_NON_MFLD_SURF')))))
= 0)))) = 0)))) = 0))) = 0;

WR12: -- legal_basis_surface FOR (pcurve and surface_curve);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_NON_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
NOT (SIZEOF (QUERY (p_oe_geom <* QUERY (oe <*
elp_fbnds.bound\path.edge_list |
SIZEOF (['AIC_NON_MFLD_SURF.PCURVE',
'AIC_NON_MFLD_SURF.SURFACE_CURVE'] *
TYPEOF (oe.edge_element.edge_geometry)) = 1 |
NOT (SIZEOF (['AIC_NON_MFLD_SURF.B_SPLINE_SURFACE',
'AIC_NON_MFLD_SURF.ELEMENTARY_SURFACE',
'AIC_NON_MFLD_SURF.OFFSET_SURFACE',
'AIC_NON_MFLD_SURF.SURFACE_REPLICA', 'AIC_NON_MFLD_SURF.SWEPT_SURFACE'] *
TYPEOF (p_oe_geom.basis_surface)) = 1)))) =
= 0 )))) = 0 ))) = 0 )))) = 0 ;

WR13: -- legal_basis_curve_for_swept_surface FOR (swept_surface);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |

```

```

NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (f_sf <* QUERY (fa <* cfs.cfs_faces |
'AIC_NON_MFLD_SURF.FACE_SURFACE' IN TYPEOF (fa)) |
NOT (('AIC_NON_MFLD_SURFADVANCED_FACE' IN TYPEOF (f_sf))
OR
(SIZEOF (QUERY (sw_sf <* f_sf.face_geometry |
'AIC_NON_MFLD_SURF.SWEPT_SURFACE' IN TYPEOF (sw_sf)) |
NOT (SIZEOF (['AIC_NON_MFLD_SURF.B_SPLINE_CURVE',
'AIC_NON_MFLD_SURF.CONIC', 'AIC_NON_MFLD_SURF.CURVE_REPLICA',
'AIC_NON_MFLD_SURF.LINE', 'AIC_NON_MFLD_SURF.OFFSET_CURVE_3D',
'AIC_NON_MFLD_SURF.PCURVE', 'AIC_NON_MFLD_SURF.POLYLINE',
'AIC_NON_MFLD_SURF.SURFACE_CURVE'] * TYPEOF (sw_sf.swept_curve))
= 1))) = 0))) = 0))) = 0;

WR14: -- mandatory_vertex_geometry FOR (vertex);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT (('AIC_NON_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_NON_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds)) |
NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
NOT ('AIC_NON_MFLD_SURF.VERTEX_POINT' IN TYPEOF
(oe.edge_element.edge_start)
AND
'AIC_NON_MFLD_SURF.VERTEX_POINT' IN TYPEOF (oe.edge_element.edge_end)))) =
0))) = 0))) = 0))) = 0;

WR15: -- legal_vertex_geometry FOR (vertex_point);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT (('AIC_NON_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_NON_MFLD_SURF.EDGE_LOOP' IN TYPEOF (bnds.bound)) |
NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
NOT (SIZEOF (['AIC_NON_MFLD_SURFCARTESIAN_POINT',
'AIC_NON_MFLD_SURF.DEGENERATE_PCURVE',
'AIC_NON_MFLD_SURF.POINT_ON_CURVE',
'AIC_NON_MFLD_SURF.POINT_ON_SURFACE'] * TYPEOF
(oe.edge_element.edge_start)) = 1
AND

```

```

(SIZEOF (['AIC_NON_MFLD_SURF.CARTESIAN_POINT',
'AIC_NON_MFLD_SURF.DEGENERATE_PCURVE',
'AIC_NON_MFLD_SURF.POINT_ON_CURVE',
'AIC_NON_MFLD_SURF.POINT_ON_SURFACE'] * TYPEOF
(oe.edge_element.edge_end)) = 1)))
= 0))) = 0))) = 0))) = 0;

WR16: -- mandatory_vertex_geometry_in_vertex_loop FOR (vertex);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT ('AIC_NON_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_NON_MFLD_SURF.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
NOT ('AIC_NON_MFLD_SURF.VERTEX_POINT' IN TYPEOF
(vlp_fbnds.loop_vertex)))) = 0)))) = 0))) = 0))) = 0;

WR17: -- legal_vertex_geometry_in_vertex_loop FOR (vertex_point);
SIZEOF (QUERY (fbsm <* QUERY (it <* items |
'AIC_NON_MFLD_SURF.FACE_BASED_SURFACE_MODEL' IN TYPEOF (it)) |
NOT (SIZEOF (QUERY (cfs <* fbsm.fbsm_faces |
NOT (SIZEOF (QUERY (fa <* cfs.cfs_faces |
NOT ('AIC_NON_MFLD_SURFADVANCED_FACE' IN TYPEOF (fa))
OR
(SIZEOF (QUERY (vlp_fbnds <* QUERY (bnds <* fa.bounds |
'AIC_NON_MFLD_SURF.VERTEX_LOOP' IN TYPEOF (bnds.bound)) |
NOT (SIZEOF (['AIC_NON_MFLD_SURF.CARTESIAN_POINT',
'AIC_NON_MFLD_SURF.DEGENERATE_PCURVE',
'AIC_NON_MFLD_SURF.POINT_ON_CURVE',
'AIC_NON_MFLD_SURF.POINT_ON_SURFACE'] * TYPEOF
(vlp_fbnds.loop_vertex)) = 1))) = 0)))) = 0))) = 0))) = 0;

END_ENTITY; -- non_manifold_surface_shape_representation

ENTITY offset_curve_3d
SUBTYPE OF (curve);
basis_curve : curve;
distance : length_measure;
self_intersect : LOGICAL;
ref_direction : direction;
WHERE
wr1: (basis_curve.dim = 3) AND (ref_direction.dim = 3);
END_ENTITY; -- offset_curve_3d

```

```

ENTITY offset_surface
  SUBTYPE OF (surface);
    basis_surface : surface;
    distance      : length_measure;
    self_intersect : LOGICAL;
END_ENTITY; -- offset_surface

ENTITY open_shell
  SUBTYPE OF (connected_face_set);
END_ENTITY; -- open_shell

ENTITY oriented_closed_shell
  SUBTYPE OF (closed_shell);
    closed_shell_element : closed_shell;
    orientation          : BOOLEAN;
DERIVE
  cfs_faces : SET [1:?] OF face := conditional_reverse(SELF.
              orientation,SELF.closed_shell_element.cfs_faces);
WHERE
  wr1: NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.ORIENTED_CLOSED_SHELL'
             IN TYPEOF(SELF.closed_shell_element));
END_ENTITY; -- oriented_closed_shell

ENTITY oriented_edge
  SUBTYPE OF (edge);
    edge_element : edge;
    orientation   : BOOLEAN;
DERIVE
  edge_start : vertex := boolean_choose(SELF.orientation,SELF.
              edge_element.edge_start,SELF.edge_element.edge_end);
  edge_end   : vertex := boolean_choose(SELF.orientation,SELF.
              edge_element.edge_end,SELF.edge_element.edge_start);
WHERE
  wr1: NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.ORIENTED_EDGE' IN
             TYPEOF(SELF.edge_element));
END_ENTITY; -- oriented_edge

ENTITY oriented_face
  SUBTYPE OF (face);
    face_element : face;
    orientation   : BOOLEAN;
DERIVE
  bounds : SET [1:?] OF face_bound := conditional_reverse(SELF.
              orientation,SELF.face_element.bounds);
WHERE
  wr1: NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.ORIENTED_FACE' IN

```

```

        TYPEOF(SELF.face_element));
END_ENTITY; -- oriented_face

ENTITY oriented_open_shell
  SUBTYPE OF (open_shell);
    open_shell_element : open_shell;
    orientation        : BOOLEAN;
DERIVE
  cfs_faces : SET [1:?] OF face := conditional_reverse(SELF.
              orientation,SELF.open_shell_element.cfs_faces);
WHERE
  wr1: NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.ORIENTED_OPEN_SHELL' IN
             TYPEOF(SELF.open_shell_element));
END_ENTITY; -- oriented_open_shell

ENTITY oriented_path
  SUBTYPE OF (path);
    path_element : path;
    orientation   : BOOLEAN;
DERIVE
  edge_list : LIST [1:?] OF UNIQUE oriented_edge :=
              conditional_reverse(SELF.orientation,SELF.path_element
              .edge_list);
WHERE
  wr1: NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.ORIENTED_PATH' IN
             TYPEOF(SELF.path_element));
END_ENTITY; -- oriented_path

ENTITY outer_boundary_curve
  SUBTYPE OF (boundary_curve);
END_ENTITY; -- outer_boundary_curve

ENTITY parabola
  SUBTYPE OF (conic);
    focal_dist : length_measure;
WHERE
  wr1: focal_dist <> 0;
END_ENTITY; -- parabola

ENTITY parametric_representation_context
  SUBTYPE OF (geometric_representation_context);
END_ENTITY; -- parametric_representation_context

ENTITY path
  SUPERTYPE OF (ONEOF (edge_loop,oriented_path))
  SUBTYPE OF (topological_representation_item);

```

```

    edge_list : LIST [1:?] OF UNIQUE oriented_edge;
  WHERE
    wr1: path_head_to_tail(SELF);
  END_ENTITY; -- path

ENTITY pcurve
  SUBTYPE OF (curve);
    basis_surface      : surface;
    reference_to_curve : definitional_representation_item;
  WHERE
    wr1: SIZEOF(reference_to_curve.items) = 1;
    wr2: 'MECHANICAL_DESIGN_SURFACE_SCHEMA.CURVE' IN TYPEOF(
      reference_to_curve.items[1]);
    wr3: reference_to_curve.items[1]\geometric_representation_item.dim =
      2;
  END_ENTITY; -- pcurve

ENTITY placement
  SUPERTYPE OF (ONEOF (axis1_placement, axis2_placement_2d,
    axis2_placement_3d))
  SUBTYPE OF (geometric_representation_item);
    location : cartesian_point;
  END_ENTITY; -- placement

ENTITY planar_box
  SUBTYPE OF (planar_extent, curve);
    placement : axis2_placement;
  END_ENTITY; -- planar_box

ENTITY planar_extent
  SUBTYPE OF (geometric_representation_item);
    size_in_x : length_measure;
    size_in_y : length_measure;
  END_ENTITY; -- planar_extent

ENTITY plane
  SUBTYPE OF (elementary_surface);
  END_ENTITY; -- plane

ENTITY plane_angle_measure_with_unit
  SUBTYPE OF (measure_with_unit);
  WHERE
    wr1: TYPEOF(SELF.unit_component) =
      'MECHANICAL_DESIGN_SURFACE_SCHEMA.PLANE_ANGLE_UNIT';
  END_ENTITY; -- plane_angle_measure_with_unit

```

```

ENTITY plane_angle_unit
  SUBTYPE OF (named_unit);
  WHERE
    wr1: (((((SELF.dimensions.length_exponent = 0) AND (SELF.dimensions
      .mass_exponent = 0)) AND (SELF.dimensions.time_exponent = 0)) AND (
      SELF.dimensions.electric_current_exponent = 0)) AND (
      SELF.dimensions.thermodynamic_temperature_exponent = 0)) AND (
      (SELF.dimensions.amount_of_substance_exponent = 0)) AND (
      SELF.dimensions.luminous_intensity_exponent = 0);
END_ENTITY; -- plane_angle_unit

ENTITY point
  SUPERTYPE OF (ONEOF (cartesian_point,point_on_curve,point_on_surface,
    degenerate_pcurve,point_replica))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY; -- point

ENTITY point_on_curve
  SUBTYPE OF (point);
  basis_curve : curve;
  point_parameter : parameter_value;
END_ENTITY; -- point_on_curve

ENTITY point_on_surface
  SUBTYPE OF (point);
  basis_surface : surface;
  point_parameter_u : parameter_value;
  point_parameter_v : parameter_value;
END_ENTITY; -- point_on_surface

ENTITY point_replica
  SUBTYPE OF (point);
  parent_pt : point;
  transformation : cartesian_transformation_operator;
  WHERE
    wr1: transformation.dim = parent_pt.dim;
END_ENTITY; -- point_replica

ENTITY point_style;
  marker : marker_select;
  marker_size : size_select;
  marker_colour : colour;
END_ENTITY; -- point_style

ENTITY polyline
  SUBTYPE OF (bounded_curve);

```

```

        points : LIST [2:?] OF cartesian_point;
END_ENTITY; -- polyline

ENTITY pre_defined_item;
    name : label;
END_ENTITY; -- pre_defined_item

ENTITY pre_defined_presentation_style
    SUBTYPE OF (pre_defined_item);
END_ENTITY; -- pre_defined_presentation_style

ENTITY pre_defined_text_font
    SUBTYPE OF (pre_defined_item);
END_ENTITY; -- pre_defined_text_font

ENTITY presentation_area
    SUBTYPE OF (presentation_representation);
WHERE
    wr1: (SIZEOF(USEDIN(SELF,('MECHANICAL_DESIGN_SURFACE_SCHEMA.' +
        'PRESENTATION REPRESENTATION_RELATIONSHIP.')) +
        'PARENT REPRESENTATION')) > 0) OR (SIZEOF(QUERY ( item <*
        SELF\representation.items | (
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.MAPPED_ITEM' IN TYPEOF(
                item)) )) > 0);
    wr2: ('MECHANICAL_DESIGN_SURFACE_SCHEMA.AREA_IN_SET' IN TYPEOF(SELF))
        OR (SIZEOF(USEDIN(SELF,'MECHANICAL_DESIGN_SURFACE_SCHEMA.' +
            'PRESENTATION_SIZE.UNIT')) = 1);
    wr3: SIZEOF(QUERY ( item <* SELF\representation.items | (((
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.MAPPED_ITEM' IN TYPEOF(
                item)) AND (SIZEOF([
                    'MECHANICAL_DESIGN_SURFACE_SCHEMA.PRESENTATION_AREA',
                    'MECHANICAL_DESIGN_SURFACE_SCHEMA.PRESENTATION_VIEW',
                    'MECHANICAL_DESIGN_SURFACE_SCHEMA.' +
                    'AREA_DEPENDENT_ANNOTATION_REPRESENTATION']) * TYPEOF(item\
                    mapped_item.mapping_source.mapped_representation)) = 0))) ) )
        = 0;
END_ENTITY; -- presentation_area

ENTITY presentation_layer_assignment;
    layer_value : identifier;
    layered_representation : SET [1:?] OF layered_item;
END_ENTITY; -- presentation_layer_assignment

ENTITY presentation_layer_usage;
    assignment : presentation_layer_assignment;
    representation : presentation_representation;

```

```

    visibility      : LOGICAL;
UNIQUE
    ur1 : assignment, representation;
END_ENTITY; -- presentation_layer_usage

ENTITY presentation_representation
    SUBTYPE OF (representation);
    WHERE
        wr1: SELF\representation.context_of_items\
            geometric_representation_context.coordinate_space_dimension
            = 2;
        wr2: 'MECHANICAL_DESIGN_SURFACE_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT'
            IN TYPEOF(SELF\representation.context_of_items);
END_ENTITY; -- presentation_representation

ENTITY presentation_size;
    unit : presentation_size_assignment_select;
    size : planar_box;
    WHERE
        wr1: (('MECHANICAL_DESIGN_SURFACE_SCHEMA.PRESENTATION REPRESENTATION'
            IN TYPEOF(SELF.unit)) AND item_in_context(SELF.size,SELF.
            unit\representation.context_of_items)) OR ((((
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.SIZE_OF_AREA_IN_SET' IN
            TYPEOF(SELF.unit)) AND (SIZEOF(QUERY ( area <* SELF.unit\
            size_of_area_in_set.of_set.areas | (NOT item_in_context(SELF
            .size,area.context_of_items)) )) = 0));
END_ENTITY; -- presentation_size

ENTITY presentation_style_assignment;
    styles : SET [1:?] OF presentation_style_select;
    WHERE
        wr1: SIZEOF(QUERY ( style1 <* SELF.styles | (SIZEOF(
            QUERY ( style2 <* (SELF.styles - style1) | ((TYPEOF(style1)
            = TYPEOF(style2)) AND ((SIZEOF([
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.' + 'SURFACE_STYLE_USAGE',
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.' +
            'EXTERNALLY_DEFINED_STYLE'] * TYPEOF(style1)) = 0) XOR ((((
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.SURFACE_STYLE_USAGE' IN
            TYPEOF(style1)) AND ((style1.side = style2.side) OR (style1.
            side = both))))))) > 0) )) = 0;
END_ENTITY; -- presentation_style_assignment

ENTITY presentation_style_by_context
    SUBTYPE OF (presentation_style_assignment);
    style_context : style_context_select;
END_ENTITY; -- presentation_style_by_context

```

```

ENTITY presentation_view
  SUBTYPE OF (presentation_representation);
  WHERE
    wr1: (SIZEOF(USEDIN(SELF,(‘MECHANICAL DESIGN_SURFACE_SCHEMA.’ +
      ‘PRESENTATION REPRESENTATION_RELATIONSHIP.’) +
      ‘PARENT REPRESENTATION’)) > 0) OR (SIZEOF(QUERY ( item <*
      SELF\representation.items | (
        ‘MECHANICAL DESIGN_SURFACE_SCHEMA.MAPPED_ITEM’ IN TYPEOF(
        item)) )) > 0);
    wr2: SIZEOF(QUERY ( item <* SELF\representation.items | (((
      ‘MECHANICAL DESIGN_SURFACE_SCHEMA.MAPPED_ITEM’ IN TYPEOF(
      item)) AND (SIZEOF([‘MECHANICAL DESIGN_SURFACE_SCHEMA.’ +
        ‘PRODUCT DATA REPRESENTATION_VIEW’,
        ‘MECHANICAL DESIGN_SURFACE_SCHEMA.’ +
        ‘VIEW_DEPENDENT_ANNOTATION REPRESENTATION’] * TYPEOF(item\
        mapped_item.mapping_source.mapped_representation)) = 0))) ) )
      = 0;
  END_ENTITY; -- presentation_view

ENTITY product;
  id : identifier;
  name : label;
  description : text;
  frame_of_reference : product_context;
  UNIQUE
  ur1 : id;
END_ENTITY; -- product

ENTITY product_category
  SUPERTYPE OF (product_related_product_category);
  name : label;
  description : OPTIONAL text;
END_ENTITY; -- product_category

ENTITY product_related_product_category
  SUBTYPE OF (product_category);
  products : SET [1:?] OF product;
END_ENTITY; -- product_related_product_category

ENTITY product_context
  SUBTYPE OF (application_context_element);
  discipline_type : label;
END_ENTITY; -- product_context

ENTITY product_data_representation_view

```

```

SUBTYPE OF (presentation_representation);
WHERE
    wr1: (SIZEOF(SELF\representation.items) = 1) AND (
        'MECHANICAL DESIGN_SURFACE_SCHEMA.CAMERA_IMAGE' IN TYPEOF(
            SELF\representation.items[1]));
END_ENTITY; -- product_data_representation_view

ENTITY product_data_representation_view_with_hlhsr
    SUBTYPE OF (product_data_representation_view);
    hidden_line_surface_removal : BOOLEAN;
END_ENTITY; -- product_data_representation_view_with_hlhsr

ENTITY product_definition;
    description : text;
    version : product_version;
    frame_of_reference : product_definition_context;
END_ENTITY; -- product_definition

ENTITY product_definition_context
    SUBTYPE OF (application_context_element);
    life_cycle_stage : label;
END_ENTITY; -- product_definition_context

ENTITY product_definition_relationship;
    id : identifier;
    name : label;
    description : text;
    relating_product_definition : product_definition;
    related_product_definition : product_definition;
END_ENTITY; -- product_definition_relationship

ENTITY product_definition_shape;
    name : label;
    description : text;
    definition : characterized_product_definition;
    UNIQUE
    ur1 : definition;
END_ENTITY; -- product_definition_shape

ENTITY product_definition_usage
    SUPERTYPE OF (assembly_component_usage)
    SUBTYPE OF (product_definition_relationship);
    UNIQUE
    ur1 : id, relating_product_definition, related_product_definition;
    WHERE
        wr1: acyclic_product_definition_relationship(SELF,[SELF\

```

```

        product_definition_relationship.related_product_definition],
        'PRODUCT_STRUCTURE.PRODUCT_DEFINITION_USAGE.' +
        'RELATED_PRODUCT_DEFINITION');
END_ENTITY; -- product_definition_usage

ENTITY product_version;
  id          : identifier;
  description : text;
  of_product  : product;
  UNIQUE
  ur1 : id, of_product;
END_ENTITY; -- product_version

ENTITY quasi_uniform_curve
  SUBTYPE OF (b_spline_curve);
END_ENTITY; -- quasi_uniform_curve

ENTITY quasi_uniform_surface
  SUBTYPE OF (b_spline_surface);
  DERIVE
    ku_up : INTEGER := (u_upper - u_degree) + 2;
    kv_up : INTEGER := (v_upper - v_degree) + 2;
END_ENTITY; -- quasi_uniform_surface

ENTITY rational_b_spline_curve
  SUBTYPE OF (b_spline_curve);
  weights_data : LIST [2:?] OF REAL;
  DERIVE
    weights : ARRAY [0:upper_index_on_control_points] OF REAL :=
      list_to_array(weights_data,0,
                    upper_index_on_control_points);
  WHERE
    wr1: SIZEOF(weights_data) = SIZEOF(SELF\b_spline_curve.
                                         control_points_list);
    wr2: curve_weights_positive(SELF);
END_ENTITY; -- rational_b_spline_curve

ENTITY rational_b_spline_surface
  SUBTYPE OF (b_spline_surface);
  weights_data : LIST [2:?] OF LIST [2:?] OF REAL;
  DERIVE
    weights : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF REAL :=
      make_array_of_array(weights_data,0,u_upper,0,v_upper);
  WHERE
    wr1: (SIZEOF(weights_data) = SIZEOF(SELF\b_spline_surface.
                                         control_points_list)) AND (SIZEOF(weights_data[1]) = SIZEOF(

```

```

        SELF\b_spline_surface.control_points_list[1]));
wr2: surface_weights_positive(SELF);
END_ENTITY; -- rational_b_spline_surface

ENTITY rectangular_composite_surface
SUBTYPE OF (bounded_surface);
segments : LIST [1:?] OF LIST [1:?] OF surface_patch;
DERIVE
  n_u : INTEGER := SIZEOF(segments);
  n_v : INTEGER := SIZEOF(segments[1]);
WHERE
  wr1: constraints_rect_comp_surface(SELF);
END_ENTITY; -- rectangular_composite_surface

ENTITY rectangular_trimmed_surface
SUBTYPE OF (bounded_surface);
basis_surface : surface;
u1           : parameter_value;
u2           : parameter_value;
v1           : parameter_value;
v2           : parameter_value;
usense       : BOOLEAN;
vsense       : BOOLEAN;
WHERE
  wr1: u1 <> u2;
  wr2: v1 <> v2;
  wr3: (((('MECHANICAL_DESIGN_SURFACE_SCHEMA.ELEMENTARY_SURFACE' IN
            TYPEOF(basis_surface)) AND ((NOT
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.PLANE') IN TYPEOF(
            basis_surface))) OR (
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.SURFACE_OF_REVOLUTION' IN
            TYPEOF(basis_surface))) OR (usense = (u2 > u1));
  wr4: ((('MECHANICAL_DESIGN_SURFACE_SCHEMA.SPHERICAL_SURFACE' IN
            TYPEOF(basis_surface)) OR (
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.TOROIDAL_SURFACE' IN
            TYPEOF(basis_surface))) OR (vsense = (v2 > v1)));
END_ENTITY; -- rectangular_trimmed_surface

ENTITY reparametrised_composite_curve_segment
SUBTYPE OF (composite_curve_segment);
param_length : parameter_value;
WHERE
  wr1: param_length > 0;
END_ENTITY; -- reparametrised_composite_curve_segment

ENTITY representation;

```

```

        items           : SET [1:?] OF representation_item;
        context_of_items : representation_context;
END_ENTITY; -- representation

ENTITY representation_context;
    context_identifier : identifier;
    context_type       : text;
    INVERSE
        contexted_representations          : SET OF representation FOR
                                              context_of_items;
        contexted_definitional_representation_items : SET OF definitional_representation_item FOR
                                                      context_of_items;
    WHERE
        wr1: (SIZEOF(SELF.contexted_representations) + SIZEOF(SELF.
                      contexted_definitional_representation_items)) > 0;
END_ENTITY; -- representation_context

ENTITY representation_dependent_styled_item
    SUBTYPE OF (styled_item);
    rep_using_item : representation;
    WHERE
        wr1: SELF.rep_using_item IN using_representations(SELF\styled_item.
                  item);
END_ENTITY; -- representation_dependent_styled_item

ENTITY representation_item;
    WHERE
        wr1: (SIZEOF(using_representations(SELF)) + SIZEOF(
                  using_definitional_representation_items(SELF))) > 0;
END_ENTITY; -- representation_item

ENTITY representation_item_without_style
    SUBTYPE OF (representation_item);
    WHERE
        wr1: SIZEOF(USEDIN(SELF,'MECHANICAL DESIGN_SURFACE_SCHEMA.' +
                  'STYLED_ITEM')) = 0;
END_ENTITY; -- representation_item_without_style

ENTITY representation_map;
    mapping_origin      : representation_item;
    mapped_representation : representation;
    INVERSE
        map_usage : SET [1:?] OF mapped_item FOR mapping_source;
    WHERE
        wr1: item_in_context(SELF.mapping_origin,SELF.mapped_representation.
                  context_of_items);

```

```

END_ENTITY; -- representation_map

ENTITY representation_relationship;
    rep_1 : representation;
    rep_2 : representation;
END_ENTITY; -- representation_relationship

ENTITY representation_relationship_with_transformation
    SUBTYPE OF (representation_relationship);
        transformation_operator : transformation;
    WHERE
        wr1: SELF\representation_relationship.rep_1.context_of_items <> SELF
            \representation_relationship.rep_2.context_of_items;
END_ENTITY; -- representation_relationship_with_transformation

ENTITY seam_curve
    SUBTYPE OF (surface_curve);
    WHERE
        wr1: SIZEOF(SELF\surface_curve.associated_geometry) = 2;
        wr2: associated_surface(associated_geometry[1]) =
            associated_surface(associated_geometry[2]);
        wr3: 'MECHANICAL DESIGN_SURFACE_SCHEMA.PCURVE' IN TYPEOF(
            associated_geometry[1]);
        wr4: 'MECHANICAL DESIGN_SURFACE_SCHEMA.PCURVE' IN TYPEOF(
            associated_geometry[2]);
END_ENTITY; -- seam_curve

ENTITY shape_aspect;
    name          : label;
    description   : text;
    of_shape     : product_definition_shape;
    product_definitional : LOGICAL;
END_ENTITY; -- shape_aspect

ENTITY shape_definition_representation;
    representation_model : shape_representation;
    representation_of   : shape_definition;
END_ENTITY; -- shape_definition_representation

ENTITY shape_representation
    SUBTYPE OF (representation);
END_ENTITY; -- shape_representation

ENTITY shape_representation_relationship
    SUBTYPE OF (representation_relationship);
    WHERE

```

```

wr1: 'MECHANICAL_DESIGN_SURFACE_SCHEMA.SHAPE REPRESENTATION' IN (
    TYPEOF(SELF\representation_relationship.rep_1) OR TYPEOF(
        SELF\representation_relationship.rep_2));
END_ENTITY; -- shape_representation_relationship

ENTITY shell_based_surface_model
    SUBTYPE OF (geometric_representation_item);
    sbsm_boundary : SET [1:?] OF shell;
    WHERE
        wr1: constraints_geometry_sb_surface_model(SELF);
    END_ENTITY; -- shell_based_surface_model

ENTITY si_unit
    SUBTYPE OF (named_unit);
    prefix : OPTIONAL si_prefix;
    name   : si_unit_name;
    DERIVE
        dimensions : dimensional_exponents := dimensions_for_si_unit(SELF.
            name);
    END_ENTITY; -- si_unit

ENTITY spherical_surface
    SUBTYPE OF (elementary_surface);
    radius : positive_length_measure;
    END_ENTITY; -- spherical_surface

ENTITY styled_item
    SUBTYPE OF (representation_item);
    styles : SET [1:?] OF presentation_style_assignment;
    item   : representation_item;
    WHERE
        wr1: (SIZEOF(SELF.styles) = 1) XOR (SIZEOF(QUERY ( pres_style <*
            SELF.styles | (NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.PRESENTATION_STYLE_BY_CON
            IN TYPEOF(pres_style)))) )) = 0);
    END_ENTITY; -- styled_item

ENTITY surface
    SUPERTYPE OF (ONEOF (elementary_surface,swept_surface,bounded_surface,
        offset_surface,surface_replica))
    SUBTYPE OF (geometric_representation_item);
    END_ENTITY; -- surface

ENTITY surface_curve
    SUPERTYPE OF (ONEOF (intersection_curve,seam_curve))
    SUBTYPE OF (curve);
    curve_3d           : curve;

```

```

associated_geometry : LIST [1:2] OF pcurve_or_surface;
master_representation : preferred_surface_curve_representation;
DERIVE
  basis_surface : surface := associated_surface(associated_geometry[1]);
WHERE
  wr1: curve_3d.dim = 3;
  wr2: ('MECHANICAL_DESIGN_SURFACE_SCHEMA.PCURVE' IN TYPEOF(
    associated_geometry[1])) OR (master_representation <>
    pcurve_s1);
  wr3: ('MECHANICAL_DESIGN_SURFACE_SCHEMA.PCURVE' IN TYPEOF(
    associated_geometry[2])) OR (master_representation <>
    pcurve_s2);
  wr4: NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.PCURVE' IN TYPEOF(
    curve_3d));
END_ENTITY; -- surface_curve

ENTITY surface_of_linear_extrusion
  SUBTYPE OF (swept_surface);
  extrusion_axis : vector;
END_ENTITY; -- surface_of_linear_extrusion

ENTITY surface_of_revolution
  SUBTYPE OF (swept_surface);
  axis_position : axis1_placement;
DERIVE
  axis_line : line := line(axis_position.location, axis_position.z);
END_ENTITY; -- surface_of_revolution

ENTITY surface_patch
  SUBTYPE OF (bounded_surface);
  parent_surface : bounded_surface;
  u_transition : transition_code;
  v_transition : transition_code;
  u_sense : BOOLEAN;
  v_sense : BOOLEAN;
WHERE
  wr1: (NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.CURVE_BOUNDED_SURFACE'
    IN TYPEOF(parent_surface))) AND (NOT (
    'MECHANICAL_DESIGN_SURFACE_SCHEMA.SURFACE_PATCH' IN TYPEOF(
      parent_surface)));
END_ENTITY; -- surface_patch

ENTITY surface_rendering_properties;
  rendered_colour : colour;
END_ENTITY; -- surface_rendering_properties

```

```

ENTITY surface_replica
  SUBTYPE OF (surface);
    parent_surface : surface;
    transformation : cartesian_transformation_operator_3d;
END_ENTITY; -- surface_replica

ENTITY surface_side_style;
  styles : SET [1:7] OF surface_style_element_select;
  WHERE
    wr1: SIZEOF(QUERY ( style1 <* SELF.styles | (SIZEOF(
      QUERY ( style2 <* (SELF.styles - style1) | (TYPEOF(style1) =
        TYPEOF(style2)) ) > 0) )) = 0;
END_ENTITY; -- surface_side_style

ENTITY surface_style_boundary;
  style_of_boundary : curve_or_render;
END_ENTITY; -- surface_style_boundary

ENTITY surface_style_control_grid;
  style_of_control_grid : curve_or_render;
END_ENTITY; -- surface_style_control_grid

ENTITY surface_style_parameter_line;
  style_of_parameter_lines : curve_or_render;
  direction_counts       : SET [1:2] OF direction_count_select;
  WHERE
    wr1: (HIINDEX(SELF.direction_counts) = 1) XOR (TYPEOF(SELF.
      direction_counts[1]) <> TYPEOF(SELF.direction_counts[2]));
END_ENTITY; -- surface_style_parameter_line

ENTITY surface_style_rendering;
  rendering_method : shading_surface_method;
  surface_colour   : colour;
END_ENTITY; -- surface_style_rendering

ENTITY surface_style_segmentation_curve;
  style_of_segmentation_curve : curve_or_render;
END_ENTITY; -- surface_style_segmentation_curve

ENTITY surface_style_silhouette;
  style_of_silhouette : curve_or_render;
END_ENTITY; -- surface_style_silhouette

ENTITY surface_style_usage;
  side   : surface_side;
  style : surface_side_style_select;

```

```

END_ENTITY; -- surface_style_usage

ENTITY swept_surface
  SUPERTYPE OF (ONEOF (surface_of_linear_extrusion,surface_of_revolution))
  SUBTYPE OF (surface);
  swept_curve : curve;
END_ENTITY; -- swept_surface

ENTITY symbol_representation
  SUBTYPE OF (representation);
END_ENTITY; -- symbol_representation

ENTITY text_literal_mapped_item
  SUBTYPE OF (mapped_item);
  WHERE
    wr1: 'MECHANICAL_DESIGN_SURFACE_SCHEMA.TEXT_LITERAL_SYMBOL' IN
      TYPEOF(SELF\mapped_item.mapping_source.mapped_representation);
    wr2: USEDIN(SELF,'') = USEDIN(SELF,
      'MECHANICAL_DESIGN_SURFACE_SCHEMA.' +
      'TEXT_STRING_REPRESENTATION.STRINGS');
END_ENTITY; -- text_literal_mapped_item

ENTITY text_literal_symbol
  SUBTYPE OF (text_symbol);
  literal : presentable_text;
  origin : axis2_placement;
  alignment : text_alignment;
  path : text_path;
  font : font_select;
  DERIVE
    SELF\representation.items : SET [1:1] OF axis2_placement := [SELF.
      origin];
END_ENTITY; -- text_literal_symbol

ENTITY text_string_representation
  SUBTYPE OF (representation);
  strings : SET [1:?] OF text_or_character;
  placement : axis2_placement;
  DERIVE
    SELF\representation.items : SET [1:?] OF representation_item := SELF
      .strings + [SELF.placement];
  WHERE
    wr1: SIZEOF(USEDIN(SELF,'MECHANICAL_DESIGN_SURFACE_SCHEMA.' +
      'ANNOTATION_TEXT_MAP.TEXT_STRING')) > 0;
    wr2: SIZEOF(QUERY ( str <* SELF.strings | (NOT ((((
      'MECHANICAL_DESIGN_SURFACE_SCHEMA.' +

```

```

        'TEXT_LITERAL_MAPPED_ITEM') IN TYPEOF(str)) AND (str.
            mapping_target := SELF.placement))) )) = 0;
END_ENTITY; -- text_string_representation

ENTITY text_symbol
    SUBTYPE OF (symbol_representation);
END_ENTITY; -- text_symbol

ENTITY topological_representation_item
    SUPERTYPE OF (ONEOF (vertex,edge,path,loop,face_bound,face,
        connected_face_set))
    SUBTYPE OF (representation_item);
END_ENTITY; -- topological_representation_item

ENTITY toroidal_surface
    SUBTYPE OF (elementary_surface);
    major_radius : positive_length_measure;
    minor_radius : positive_length_measure;
WHERE
    wr1: major_radius > minor_radius;
END_ENTITY; -- toroidal_surface

ENTITY trimmed_curve
    SUBTYPE OF (bounded_curve);
    basis_curve          : curve;
    trim_1                : SET [1:2] OF trimming_select;
    trim_2                : SET [1:2] OF trimming_select;
    sense_agreement       : BOOLEAN;
    master_representation : trimming_preference;
WHERE
    wr1: (HIINDEX(trim_1) = 1) XOR (TYPEOF(trim_1[1]) <> TYPEOF(trim_1[2]));
    wr2: (HIINDEX(trim_2) = 1) XOR (TYPEOF(trim_2[1]) <> TYPEOF(trim_2[2]));
END_ENTITY; -- trimmed_curve

ENTITY uniform_curve
    SUBTYPE OF (b_spline_curve);
END_ENTITY; -- uniform_curve

ENTITY uniform_surface
    SUBTYPE OF (b_spline_surface);
END_ENTITY; -- uniform_surface

ENTITY vector
    SUBTYPE OF (geometric_representation_item);
    orientation : direction;
    magnitude   : length_measure;

```

```

WHERE
    wr1: magnitude >= 0;
END_ENTITY; -- vector

ENTITY vertex
    SUBTYPE OF (topological_representation_item);
END_ENTITY; -- vertex

ENTITY vertex_loop
    SUBTYPE OF (loop);
    loop_vertex : vertex;
END_ENTITY; -- vertex_loop

ENTITY vertex_point
    SUBTYPE OF (vertex, geometric_representation_item);
    vertex_geometry : point;
END_ENTITY; -- vertex_point

ENTITY view_dependent_annotation_representation
    SUBTYPE OF (presentation_representation);
WHERE
    wr1: SIZEOF(USEDIN(SELF, ('MECHANICAL DESIGN_SURFACE_SCHEMA.' +
        'PRESENTATION REPRESENTATION_RELATIONSHIP.') +
        'PARENT REPRESENTATION')) = 0;
    wr2: SIZEOF(QUERY ( item <* SELF\representation.items | (SIZEOF([
        'MECHANICAL DESIGN_SURFACE_SCHEMA.' +
        'ANNOTATION_OCCURRENCE',
        'MECHANICAL DESIGN_SURFACE_SCHEMA.AXIS2_PLACEMENT'] *
        TYPEOF(item)) = 0) )) = 0;
END_ENTITY; -- view_dependent_annotation_representation

ENTITY view_volume;
    projection_type : central_or_parallel;
    projection_point : cartesian_point;
    view_plane_distance : length_measure;
    front_plane_distance : length_measure;
    front_plane_clipping : BOOLEAN;
    back_plane_distance : length_measure;
    back_plane_clipping : BOOLEAN;
    view_volume_sides_clipping : BOOLEAN;
    view_window : planar_box;
WHERE
    wr1: SELF.front_plane_distance < SELF.back_plane_distance;
    wr2: SELF.view_window.placement.location.coordinates[3] = SELF.
        view_plane_distance;
    wr3: SELF.projection_point.coordinates[3] <> SELF.

```

```

        view_plane_distance;
END_ENTITY; -- view_volume

RULE compatible_dimension FOR (cartesian_point, direction,
                               representation_context, geometric_representation_context);

WHERE
  wr1: SIZEOF(QUERY ( x <* cartesian_point | (SIZEOF(QUERY ( y <*
                                representation_context | (item_in_context(x,y) AND ((NOT y) IN
                                geometric_representation_context)) )) > 0) )) = 0;
  wr2: SIZEOF(QUERY ( x <* cartesian_point | (SIZEOF(QUERY ( y <*
                                geometric_representation_context | (item_in_context(x,y) AND (
                                HIINDEX(x.coordinates) <> y.coordinate_space_dimension)) )) >
                                0) )) = 0;
  wr3: SIZEOF(QUERY ( x <* direction | (SIZEOF(QUERY ( y <*
                                representation_context | (item_in_context(x,y) AND ((NOT y) IN
                                geometric_representation_context)) )) > 0) )) = 0;
  wr4: SIZEOF(QUERY ( x <* direction | (SIZEOF(QUERY ( y <*
                                geometric_representation_context | (item_in_context(x,y) AND (
                                HIINDEX(x.direction_ratios) <> y.coordinate_space_dimension)) )) >
                                0) )) = 0;

END_RULE; -- compatible_dimension

RULE coordinated_relationships FOR (shape_definition_representation);

FUNCTION relationships_are_coordinated(
  sdr: SET OF shape_definition_representation
): BOOLEAN;

LOCAL
  sr1 : shape_representation;
  sr2 : shape_representation;
  i   : INTEGER;
  j   : INTEGER;
  pd1 : product_definition;
  pd2 : product_definition;
END_LOCAL;
REPEAT i := 1 TO HIINDEX(sdr) BY 1;
  REPEAT j := 1 TO HIINDEX(sdr) BY 1;
    IF i <> j THEN
      sr1 := sdr[i].representation_model;
      sr2 := sdr[j].representation_model;
      pd1 := represented_product_definition(sdr[i]);
      pd2 := represented_product_definition(sdr[j]);
      IF (sr2 IN relatives_of_shape_representations([sr1])) XOR (pd2

```

```

        IN relatives_of_product_definitions([pd1], 'MECHANICAL_DESIGN_SURFACE_SCHEMA.PRE
        THEN
            RETURN(FALSE);
        END_IF;
        END_IF;
        END_REPEAT;
    END_REPEAT;
    RETURN(TRUE);

END_FUNCTION; -- relationships_are_coordinated

WHERE
    wr1: relationships_are_coordinated(shape_definition_representation);

END_RULE; -- coordinated_relationships

FUNCTION acyclic_mapped_representation(
    parent_set: SET OF representation;
    children_set: SET OF representation_item
): BOOLEAN;

LOCAL
    i : INTEGER;
    j : INTEGER;
    x : SET OF representation_item;
    y : SET OF representation_item;
END_LOCAL;
x := QUERY ( z <* children_set | (
    'MECHANICAL_DESIGN_SURFACE_SCHEMA.MAPPED_ITEM' IN TYPEOF(z) ) );
IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIIINDEX(x) BY 1;
        IF x[i]\mapped_item.mapping_source.mapped_representation IN
            parent_set THEN
            RETURN(FALSE);
        END_IF;
        IF NOT acyclic_mapped_representation(parent_set + x[i]\mapped_item
            .mapping_source.mapped_representation, x[i]\mapped_item.
            mapping_source.mapped_representation.items) THEN
            RETURN(FALSE);
        END_IF;
    END_REPEAT;
END_IF;
x := children_set - x;
IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIIINDEX(x) BY 1;
        y := QUERY ( z <* USEDIN(x[i], '') | (

```

```

        'MECHANICAL_DESIGN_SURFACE_SCHEMA.REPRESENTATION_ITEM' IN
        TYPEOF(z)) );
IF NOT acyclic_mapped_representation(parent_set,y) THEN
    RETURN(FALSE);
END_IF;
END_REPEAT;
END_IF;
RETURN(TRUE);

END_FUNCTION; -- acyclic_mapped_representation

FUNCTION acyclic_product_definition_relationship(
    relation: product_definition_relationship;
    relatives: SET OF product_definition;
    specific_relation: STRING
): LOGICAL;

LOCAL
    i : INTEGER;
    x : SET OF product_definition_relationship;
    local_relatives : SET OF product_definition;
END_LOCAL;
REPEAT i := 1 TO HIINDEX(relatives) BY 1;
    IF relation.relating_product_definition :=: relatives[i] THEN
        RETURN(FALSE);
    END_IF;
END_REPEAT;
x := USEDIN(relation.relating_product_definition,specific_relation);
local_relatives := relatives + relation.relating_product_definition;
IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x) BY 1;
        IF NOT acyclic_product_definition_relationship(x[i],
            local_relatives,specific_relation) THEN
            RETURN(FALSE);
        END_IF;
    END_REPEAT;
END_IF;
RETURN(TRUE);

END_FUNCTION; -- acyclic_product_definition_relationship

FUNCTION associated_surface(
    arg: pcurve_or_surface
): surface;

LOCAL

```

```

        surf : surface;
END_LOCAL;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.PCURVE' IN TYPEOF(arg) THEN
    surf := arg.basis_surface;
ELSE
    surf := arg;
END_IF;
RETURN(surf);

END_FUNCTION; -- associated_surface

FUNCTION base_axis(
    dim: INTEGER;
    axis1, axis2, axis3: direction
): LIST [2:3] OF direction;

LOCAL
    u      : LIST [2:3] OF direction;
    vec   : direction;
    factor : REAL;
END_LOCAL;
IF dim = 3 THEN
    u[3] := NVL(axis3,direction([0,0,1]));
    u[1] := first_proj_axis(u[3],axis1);
    u[2] := second_proj_axis(u[3],u[1],axis2);
ELSE
    u[3] := ?>;
    IF EXISTS(axis1) THEN
        u[1] := normalise(axis1);
        u[2] := orthogonal_complement(u[1]);
    IF EXISTS(axis2) THEN
        factor := dot_product(axis2,u[2]);
        IF factor < 0 THEN
            u[2].direction_ratios[1] := -u[2].direction_ratios[1];
            u[2].direction_ratios[2] := -u[2].direction_ratios[2];
        END_IF;
    END_IF;
ELSE
    IF EXISTS(axis2) THEN
        u[2] := normalise(axis2);
        u[1] := orthogonal_complement(u[2]);
        u[1].direction_ratios[1] := -u[1].direction_ratios[1];
        u[1].direction_ratios[2] := -u[1].direction_ratios[2];
    ELSE
        u[1].direction_ratios[1] := 1;
        u[1].direction_ratios[2] := 0;
    END_IF;
END_IF;

```

```

        u[2].direction_ratios[1] := 0;
        u[2].direction_ratios[2] := 1;
    END_IF;
    END_IF;
END_IF;
RETURN(u);

END_FUNCTION; -- base_axis

FUNCTION boolean_choose(
    b: BOOLEAN;
    choice1, choice2: GENERIC
): GENERIC;
IF b THEN
    RETURN(choice1);
ELSE
    RETURN(choice2);
END_IF;

END_FUNCTION; -- boolean_choose

FUNCTION build_2axes(
    ref_direction: direction
): LIST [2:2] OF direction;

LOCAL
    u : LIST [2:2] OF direction;
END_LOCAL;
u[1] := NVL(normalise(ref_direction),direction([1,0]));
u[2] := orthogonal_complement(u[1]);
RETURN(u);

END_FUNCTION; -- build_2axes

FUNCTION build_axes(
    axis, ref_direction: direction
): LIST [3:3] OF direction;

LOCAL
    u : LIST [3:3] OF direction;
END_LOCAL;
u[3] := NVL(normalise(axis),direction([0,0,1]));
u[1] := first_proj_axis(u[3],ref_direction);
u[2] := normalise(cross_product(u[3],u[1]));
RETURN(u);

```

```

END_FUNCTION; -- build_axes

FUNCTION build_transformed_set(
    tr: cartesian_transformation_operator;
    gset: geometric_set
): geometric_set;

LOCAL
    trcurve : curve;
    s       : SET [1:?] OF geometric_set_select := gset.elements;
    trpoint : point;
    trset   : SET [0:?] OF geometric_set_select := ?;
    trsurf  : surface;
END_LOCAL;
REPEAT j := 1 TO SIZEOF(s) BY 1;
    IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.CURVE' IN TYPEOF(s[j]) THEN
        trset := trset + curve_replica(s[j],tr);
    ELSE
        IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.POINT' IN TYPEOF(s[j]) THEN
            trset := trset + point_replica(s[j],tr);
        ELSE
            IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.SURFACE' IN TYPEOF(s[j])
                THEN
                    trset := trset + surface_replica(s[j],tr);
            END_IF;
        END_IF;
    END_IF;
END_REPEAT;
RETURN(trset);

END_FUNCTION; -- build_transformed_set

FUNCTION conditional_reverse(
    p: BOOLEAN;
    an_item: reversible_topology
): reversible_topology;
IF p THEN
    RETURN(an_item);
ELSE
    RETURN(topology_reversed(an_item));
END_IF;

END_FUNCTION; -- conditional_reverse

FUNCTION constraints_composite_curve_on_surface(
    c: composite_curve_on_surface

```

```

    ): BOOLEAN;

LOCAL
    n_segments : INTEGER := SIZEOF(c.segments);
END_LOCAL;
REPEAT k := 1 TO n_segments BY 1;
    IF ((NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.PCURVE' IN TYPEOF(c\
        composite_curve.segments[k].parent_curve))) AND (NOT (
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.SURFACE_CURVE' IN TYPEOF(c\
                composite_curve.segments[k].parent_curve)))) AND (NOT (
                    'MECHANICAL_DESIGN_SURFACE_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN
                        TYPEOF(c\composite_curve.segments[k].parent_curve))) THEN
        RETURN(FALSE);
    END_IF;
    IF get_basis_surface(c\composite_curve.segments[k].parent_curve) <>
        c.basis_surface THEN
        RETURN(FALSE);
    END_IF;
END_REPEAT;
RETURN(TRUE);

END_FUNCTION; -- constraints_composite_curve_on_surface

FUNCTION constraints_geometry_sb_surface_model(
    m: shell_based_surface_model
): BOOLEAN;

LOCAL
    result : BOOLEAN := TRUE;
END_LOCAL;
REPEAT j := 1 TO SIZEOF(m.sbsm_boundary) BY 1;
    IF (NOT ('MECHANICAL_DESIGN_SURFACE_SCHEMA.OPEN_SHELL' IN TYPEOF(m.
        sbsm_boundary[j]))) AND (NOT (
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.CLOSED_SHELL' IN TYPEOF(m.
                sbsm_boundary[j]))) THEN
        result := FALSE;
        RETURN(result);
    END_IF;
END_REPEAT;
RETURN(result);

END_FUNCTION; -- constraints_geometry_sb_surface_model

FUNCTION constraints_param_bspl(
    degree, up_knots, up_cp: INTEGER;
    knot_mult: LIST OF INTEGER;

```

```

        knots: LIST OF parameter_value
    ): BOOLEAN;

LOCAL
    k      : INTEGER;
    l      : INTEGER;
    sum   : INTEGER;
    result : BOOLEAN := TRUE;
END_LOCAL;
sum := knot_mult[1];
REPEAT i := 2 TO up_knots BY 1;
    sum := sum + knot_mult[i];
END_REPEAT;
IF (((degree < 1) OR (up_knots < 2)) OR (up_cp < degree)) OR (sum <> (
    degree + up_cp) + 2)) THEN
    result := FALSE;
    RETURN(result);
END_IF;
k := knot_mult[1];
IF (k < 1) OR (k > (degree + 1)) THEN
    result := FALSE;
    RETURN(result);
END_IF;
REPEAT i := 2 TO up_knots BY 1;
    IF (knot_mult[i] < 1) OR (knots[i] <= knots[i - 1]) THEN
        result := FALSE;
        RETURN(result);
    END_IF;
    k := knot_mult[i];
    IF (i < up_knots) AND (k > degree) THEN
        result := FALSE;
        RETURN(result);
    END_IF;
    IF (i = up_knots) AND (k > (degree + 1)) THEN
        result := FALSE;
        RETURN(result);
    END_IF;
END_REPEAT;
RETURN(result);

END_FUNCTION; -- constraints_param_bspl

FUNCTION constraints_rect_comp_surface(
    s: rectangular_composite_surface
): BOOLEAN;
REPEAT i := 1 TO s.n_u BY 1;

```

```

REPEAT j := 1 TO s.n_v BY 1;
  IF NOT (('MECHANICAL_DESIGN_SURFACE_SCHEMA.B_SPLINE_SURFACE' IN
    TYPEOF(s.segments[i][j].parent_surface)) OR (
    'MECHANICAL_DESIGN_SURFACE_SCHEMA.RECTANGULAR_TRIMMED_SURFACE'
    IN TYPEOF(s.segments[i][j].parent_surface))) THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
END_REPEAT;
REPEAT i := 1 TO s.n_u - 1 BY 1;
  REPEAT j := 1 TO s.n_v BY 1;
    IF s.segments[i][j].u_transition = discontinuous THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
END_REPEAT;
REPEAT i := 1 TO s.n_u BY 1;
  REPEAT j := 1 TO s.n_v - 1 BY 1;
    IF s.segments[i][j].v_transition = discontinuous THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
END_REPEAT;
RETURN(TRUE);

END_FUNCTION; -- constraints_rect_comp_surface

FUNCTION cross_product(
  arg1, arg2: direction
): vector;

LOCAL
  v2      : LIST [3:3] OF REAL;
  v1      : LIST [3:3] OF REAL;
  mag     : REAL;
  res    : direction;
  result : vector;
END_LOCAL;
IF ((NOT EXISTS(arg1)) OR (arg1.dim = 2)) OR ((NOT EXISTS(arg2)) OR (
  arg2.dim = 2)) THEN
  RETURN(?);
ELSE
  BEGIN
    v1 := normalise(arg1).direction_ratios;
    v2 := normalise(arg2).direction_ratios;
    res.direction_ratios[1] := (v1[2] * v2[3]) - (v1[3] * v2[2]);
  END;
END;

```

```

    res.direction_ratios[2] := (v1[3] * v2[1]) - (v1[1] * v2[3]);
    res.direction_ratios[3] := (v1[1] * v2[2]) - (v1[2] * v2[1]);
    mag := 0;
    REPEAT i := 1 TO 3 BY 1;
        mag := mag + (res.direction_ratios[i] * res.direction_ratios[i]);
    END_REPEAT;
    result.orientation := res;
    result.magnitude := SQRT(mag);
    RETURN(result);
END;
END_IF;

END_FUNCTION; -- cross_product

FUNCTION curve_weights_positive(
    b: rational_b_spline_curve
): BOOLEAN;

LOCAL
    result : BOOLEAN := TRUE;
END_LOCAL;
REPEAT i := 0 TO b.upper_index_on_control_points BY 1;
    IF b.weights[i] <= 0 THEN
        result := FALSE;
        RETURN(result);
    END_IF;
END_REPEAT;
RETURN(result);

END_FUNCTION; -- curve_weights_positive

FUNCTION derive_dimensional_exponents(
    x: unit
): dimensional_exponents;

LOCAL
    i      : INTEGER;
    result : dimensional_exponents := [];
END_LOCAL;
IF 'MECHANICAL DESIGN_SURFACE_SCHEMA.DERIVED_UNIT' IN TYPEOF(x) THEN
    REPEAT i := LOINDEX(x.elements) TO HIINDEX(x.elements) BY 1;
        result.length_exponent := result.length_exponent + (x.elements[i].
            exponent * x.elements[i].unit.dimensions.length_exponent);
        result.mass_exponent := result.mass_exponent + (x.elements[i].
            exponent * x.elements[i].unit.dimensions.mass_exponent);
        result.electric_current_exponent := result.

```

```

        electric_current_exponent + (x.elements[i].exponent * x.
        elements[i].unit.dimensions.electric_current_exponent);
    result.thermodynamic_temperature_exponent := result.
        thermodynamic_temperature_exponent + (x.elements[i].exponent * x.
        elements[i].unit.dimensions.
        thermodynamic_temperature_exponent);
    result.amount_of_substance_exponent := result.
        amount_of_substance_exponent + (x.elements[i].exponent * x.
        elements[i].unit.dimensions.amount_of_substance_exponent);
    result.luminous_intensity_exponent := result.
        luminous_intensity_exponent + (x.elements[i].exponent * x.
        elements[i].unit.dimensions.luminous_intensity_exponent);
END_REPEAT;
ELSE
    result := x.dimensions;
END_IF;
RETURN(result);

END_FUNCTION; -- derive_dimensional_exponents

FUNCTION dimension_of(
    item: geometric_representation_item
): dimension_count;

LOCAL
    x : SET OF representation;
    y : representation_context;
    z : SET OF definitional_representation_item;
END_LOCAL;
x := using_representations(item);
IF SIZEOF(x) > 0 THEN
    y := x[1].context_of_items;
    RETURN(y\geometric_representation_context.coordinate_space_dimension);
ELSE
    z := using_definitional_representation_items(item);
    y := z[1].context_of_items;
    RETURN(y\geometric_representation_context.coordinate_space_dimension);
END_IF;

END_FUNCTION; -- dimension_of

FUNCTION dimensions_for_si_unit(
    n: si_unit_name
): dimensional_exponents;
CASE n OF
    metre          : RETURN(dimensional_exponents(1,0,0,0,0,0,0));

```

```

gram      : RETURN(dimensional_exponents(0,1,0,0,0,0,0));
second    : RETURN(dimensional_exponents(0,0,1,0,0,0,0));
ampere    : RETURN(dimensional_exponents(0,0,0,1,0,0,0));
kelvin    : RETURN(dimensional_exponents(0,0,0,0,1,0,0));
mole      : RETURN(dimensional_exponents(0,0,0,0,0,1,0));
candela   : RETURN(dimensional_exponents(0,0,0,0,0,0,1));
radian    : RETURN(dimensional_exponents(0,0,0,0,0,0,0));
steradian : RETURN(dimensional_exponents(0,0,0,0,0,0,0));
hertz     : RETURN(dimensional_exponents(0,0,-1,0,0,0,0));
newton    : RETURN(dimensional_exponents(1,1,-2,0,0,0,0));
pascal    : RETURN(dimensional_exponents(-1,1,-2,0,0,0,0));
joule     : RETURN(dimensional_exponents(2,1,-2,0,0,0,0));
watt      : RETURN(dimensional_exponents(2,1,-3,0,0,0,0));
coulomb   : RETURN(dimensional_exponents(0,0,1,1,0,0,0));
volt      : RETURN(dimensional_exponents(2,1,-3,-1,0,0,0));
farad     : RETURN(dimensional_exponents(-2,-1,4,1,0,0,0));
ohm       : RETURN(dimensional_exponents(2,1,-3,-2,0,0,0));
siemens   : RETURN(dimensional_exponents(-2,-1,3,2,0,0,0));
weber     : RETURN(dimensional_exponents(2,1,-2,-1,0,0,0));
tesla     : RETURN(dimensional_exponents(0,1,-2,-1,0,0,0));
henry     : RETURN(dimensional_exponents(2,1,-2,-2,0,0,0));
degree_celsius : RETURN(dimensional_exponents(0,0,0,0,1,0,0));
lumen     : RETURN(dimensional_exponents(0,0,0,0,0,0,1));
lux       : RETURN(dimensional_exponents(-2,0,0,0,0,0,1));
becquerel : RETURN(dimensional_exponents(0,0,-1,0,0,0,0));
gray      : RETURN(dimensional_exponents(2,0,-2,0,0,0,0));
sievert   : RETURN(dimensional_exponents(2,0,-2,0,0,0,0));
END_CASE;

END_FUNCTION; -- dimensions_for_si_unit

FUNCTION dot_product(
    arg1, arg2: direction
): REAL;

LOCAL
    ndim   : INTEGER;
    scalar : REAL;
    vec1   : direction;
    vec2   : direction;
END_LOCAL;
IF (NOT EXISTS(arg1)) OR (NOT EXISTS(arg2)) THEN
    scalar := ?;
ELSE
    IF arg1.dim <> arg2.dim THEN
        scalar := ?;
    END_IF;
END_IF;

```

```

    ELSE
        BEGIN
            vec1 := normalise(arg1);
            vec2 := normalise(arg2);
            ndim := arg1.dim;
            scalar := 0;
            REPEAT i := 1 TO ndim BY 1;
                scalar := scalar + (vec1.direction_ratios[i] * vec2.
                    direction_ratios[i]);
            END_REPEAT;
        END;
        RETURN(scalar);
    END_IF;
END_IF;

END_FUNCTION; -- dot_product

FUNCTION edge_reversed(
    an_edge: oriented_edge
): edge;

LOCAL
    the_reverse : edge;
END_LOCAL;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.ORIENTED_EDGE' IN TYPEOF(an_edge)
    THEN
    the_reverse := oriented_edge(an_edge.edge_element,NOT an_edge.
        orientation);
ELSE
    the_reverse := oriented_edge(an_edge,FALSE);
END_IF;
RETURN(the_reverse);

END_FUNCTION; -- edge_reversed

FUNCTION face_bound_reversed(
    a_face_bound: face_bound
): face_bound;

LOCAL
    the_reverse : face_bound;
END_LOCAL;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF(
    a_face_bound) THEN
    the_reverse := face_outer_bound(a_face_bound.bound,NOT a_face_bound.
        orientation);

```

```

ELSE
    the_reverse := face_bound(a_face_bound.bound,NOT a_face_bound.
        orientation);
END_IF;
RETURN(the_reverse);

END_FUNCTION; -- face_bound_reversed

FUNCTION face_reversed(
    a_face: oriented_face
): face;

LOCAL
    the_reverse : face;
END_LOCAL;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.ORIENTED_FACE' IN TYPEOF(a_face)
    THEN
    the_reverse := oriented_face(a_face.face_element,NOT a_face.
        orientation);
ELSE
    the_reverse := oriented_face(a_face,FALSE);
END_IF;
RETURN(the_reverse);

END_FUNCTION; -- face_reversed

FUNCTION first_proj_axis(
    z_axis, arg: direction
): direction;

LOCAL
    v      : direction;
    x_axis : direction;
END_LOCAL;
IF (NOT EXISTS(z_axis)) OR (arg.dim <> 3) THEN
    x_axis := ?>;
ELSE
    z_axis := normalise(z_axis);
    IF NOT EXISTS(arg) THEN
        IF z_axis <> direction([1,0,0]) THEN
            v := [1,0,0];
        ELSE
            v := [0,1,0];
        END_IF;
    ELSE
        IF cross_product(arg,z_axis).magnitude = 0 THEN

```

```

        x_axis := ?;
    ELSE
        v := arg;
    END_IF;
END_IF;
x_axis := scalar_times_vector(dot_product(v,z_axis),z_axis);
x_axis := vector_diff(v,x_axis);
x_axis := normalise(x_axis);
END_IF;
RETURN(x_axis);

END_FUNCTION; -- first_proj_axis

FUNCTION gbsf_check_point (pnt : point; schema_name : STRING) :
    BOOLEAN;

-- check whether the input has the right type
IF NOT schema_name + '.POINT' IN TYPEOF (pnt) THEN
    RETURN(TRUE);
ELSE
    -- a point_on_curve needs to be checked for the validity of its curve;
    -- further references down the tree are taken care of by the function
    -- gbsf_check_curve
    IF schema_name + '.POINT_ON_CURVE' IN TYPEOF (pnt) THEN
        RETURN(gbsf_check_curve(pnt.basis_curve, schema_name));
    ELSE
        -- a point_on_surface needs to be checked for the validity of its surface
        -- further references down the tree are taken care of by the function
        -- gbsf_check_surface
        IF schema_name + '.POINT_ON_SURFACE' IN TYPEOF (pnt) THEN
            RETURN(gbsf_check_surface(pnt.basis_surface, schema_name));
        ELSE
            -- a point_on_surface needs to be checked for the validity of its surface
            -- further references down the tree are taken care of by the functions
            -- gbsf_check_curve and gbsf_check_surface;
            -- both shall return true to get a valid point
            IF schema_name + '.DEGENERATE_PCURVE' IN TYPEOF (pnt) THEN
                RETURN
            END_IF;
        END_IF;
    END_IF;
END_IF;

```

```

(gbsf_check_curve(pnt.reference_to_curve.items[1], schema_name)
AND
gbsf_check_surface(pnt.basis_surface, schema_name));
ENDIF;
ENDIF;
ENDIF;
RETURN(FALSE);

END_FUNCTION;

FUNCTION gbsf_check_curve (cv : curve; schema_name : STRING) :
BOOLEAN;

-- check whether the input has the right type
IF NOT schema_name + '.CURVE' IN TYPEOF (cv) THEN
    RETURN(TRUE);
ENDIF;

-- let those types pass that do not have any further references
-- respectively rules to be applied
IF SIZEOF ([schema_name + '.CIRCLE', schema_name + '.ELLIPSE']
* TYPEOF(cv)) = 1 THEN
    RETURN(TRUE);
ELSE

    -- the following shall not self-intersect
    IF schema_name + '.B_SPLINE_CURVE' IN TYPEOF(cv)
    AND cv.self_intersect THEN
        RETURN(TRUE);
    ELSE

        -- this references segments that are of type
        -- composite_curve_segment etc.
        IF schema_name + 'COMPOSITE_CURVE' IN TYPEOF(cv) THEN
            REPEAT i := 1 TO SIZEOF (cv.segments);
            IF NOT (gbsf_check_curve(cv.segment[i], schema_name) THEN
                RETURN(FALSE);
            END_IF;
            END_REPEAT;
            RETURN(TRUE);
        ELSE

            -- these ones have parent_curves; check them
            IF SIZEOF ([schema_name + 'CURVE_REPLICA',

```

```

schema_name + 'COMPOSITE_CURVE_SEGMENT'] * TYPEOF(cv)) = 1 THEN
    RETURN(gbsf_check_curve(cv.parent_curve, schema_name));
END_IF;

-- these ones shall be trimmed
IF SIZEOF ([schema_name + 'HYPERBOLA', schema_name + 'LINE',
schema_name + 'PARABOLA'] * TYPEOF(cv)) = 1 THEN
    IF NOT (SIZEOF (USEDIN
(cv,'AIC_GBNP_SURF.TRIMMED_CURVE.BASIS_CURVE')) = 0 ) THEN
        RETURN(TRUE);
    END_IF;
ELSE

-- offset_curve_3d references a curve and shall not self_intersect
IF schema_name + 'OFFSET_CURVE_3D' IN TYPEOF(cv) THEN
    RETURN
    (gbsf_check_curve(cv.basis_curve, schema_name)
    AND
    NOT cv.self_intersect);
ELSE

-- pcurve references a curve - indirectly, and a basis_surface
IF schema_name + 'PCURVE' IN TYPEOF(cv) THEN
    RETURN
    (gbsf_check_curve(cv.reference_to_curve.items[1], schema_name)
    AND
    gbsf_check_surface(cv.basis_surface, schema_name));
ELSE

-- polyline shall have at least 3 points and shall only
-- be used to represent an intersection_curve
IF schema_name + 'POLYLINE' IN TYPEOF(cv) THEN
    IF NOT SIZEOF (cv.points) < 3 AND
    NOT SIZEOF (USEDIN (cv,
'AIC_GBNP_SURF.INTERSECTION_CURVE.BASIS_CURVE')) = 0 ) THEN
        RETURN(TRUE);
    END_IF;
ELSE

-- surface_curve references a curve_3d and one or two pcures
-- or one or two surface_curves or one of each
IF schema_name + 'SURFACE_CURVE' IN TYPEOF(cv) THEN
    -- if the curve reference is correct, check also the rest
    IF gbsf_check_curve(cv.curve_3d, schema_name) THEN
        REPEAT i := 1 TO SIZEOF (cv.associated_geometry);
            -- do for one or two associated_geometries:

```

```

        IF schema_name + 'SURFACE' IN TYPEOF
        (cv.associated_geometry[i]) THEN
            IF NOT gbsf_check_surface(cv.associated_geometry[i] ,
            schema_name) THEN
                RETURN(FALSE);
            END_IF;
        ELSE

            IF schema_name + 'PCURVE' IN TYPEOF
            (cv.associated_geometry[i]) THEN
                IF NOT gbsf_check_curve(cv.associated_geometry[i] ,
                schema_name) THEN
                    RETURN(FALSE);
                END_IF;
                END_IF;
            END_IF;
            END_REPEAT;
            RETURN(TRUE);
        END_IF;
    ELSE

        -- trimmed_curve references a basis_curve
        IF schema_name + 'TRIMMED_CURVE' IN TYPEOF(cv) THEN
            RETURN(gbsf_check_curve(cv.basis_curve, schema_name));
        END_IF;
        END_IF;
    END_IF;

    RETURN(FALSE);
END_FUNCTION;

FUNCTION gbsf_check_surface (sf : surface; schema_name : STRING) :
BOOLEAN;

-- check whether the input has the right type
IF NOT schema_name + '.SURFACE' IN TYPEOF (sf) THEN
    RETURN(TRUE);
END_IF;

-- b_spline_surface has a self_intersect attribute that shall be false

```

```

IF schema_name + 'B_SPLINE_SURFACE' IN TYPEOF(sf) THEN
    RETURN(NOT sf.self_intersect);
ELSE

    -- these three shall be trimmed
    IF SIZEOF ([schema_name + 'CONICAL_SURFACE',
    schema_name + 'CYLINDRICAL_SURFACE', schema_name + 'PLANE']
    * TYPEOF(sf)) = 1 THEN
        IF NOT SIZEOF (USEDIN (sf,
        'AIC_GBND_SURF.RECTANGULAR_TRIMMED_SURFACE.BASIS_SURFACE') +
        USEDIN (sf, 'AIC_GBND_SURF.CURVE_BOUNDED_SURFACE.BASIS_SURFACE')) =
        0 THEN
            RETURN(TRUE);
        END_IF;
    ELSE

        --
        IF schema_name + 'CURVE_BOUNDED_SURFACE' IN TYPEOF(sf) THEN
            -- if the surface reference is correct, check the curves
            IF gbsf_check_surface(sf.basis_surface, schema_name) THEN
                REPEAT i := 1 TO SIZEOF (sf.boundaries);
                    IF NOT (gbsf_check_curve(sf.boundaries[i], schema_name) THEN
                        RETURN(FALSE);
                    END_IF;
                END_REPEAT;
                RETURN(TRUE);
            END_IF;
        ELSE

            -- offset_surface references a surface and shall not self_intersect
            IF schema_name + 'OFFSET_SURFACE' IN TYPEOF(sf) THEN
                RETURN
                (gbsf_check_surface(sf.basis_surface, schema_name)
                AND
                NOT sf.self_intersect);
            ELSE

                -- rectangular_composite_surface references a matrix of surfaces
                IF schema_name + 'RECTANGULAR_COMPOSITE_SURFACE' IN TYPEOF(sf) THEN
                    REPEAT i := 1 TO SIZEOF (sf.segments);
                        REPEAT j := 1 TO SIZEOF (sf.segments[i]);
                            IF NOT (gbsf_check_surface(sf.segments[i][j], schema_name) THEN
                                RETURN(FALSE);
                            END_IF;
                        END_REPEAT;
                    END_REPEAT;
                END_REPEAT;
            END_IF;
        END_IF;
    END_IF;

```

```

        RETURN(TRUE);
    ELSE

        -- rectangular_trimmed_surface has a basis_surface
        IF schema_name + 'RECTANGULAR_TRIMMED_SURFACE' IN TYPEOF(sf) THEN
            RETURN(gbsf_check_surface(sf.basis_surface, schema_name));
        ELSE

            -- two without any action!
            IF SIZEOF ([schema_name + 'SPHERICAL_SURFACE',
            schema_name + 'TOROIDAL_SURFACE'] * TYPEOF(sf)) = 1 THEN
                RETURN(TRUE);
            ELSE

                -- two that have parent_surfaces
                IF SIZEOF ([schema_name + 'SURFACE_PATCH',
                schema_name + 'SURFACE_REPLICA'] * TYPEOF(sf)) = 1 THEN
                    RETURN(gbsf_check_surface(sf.parent_surface, schema_name));
                ELSE

                    -- and the swept_surface with its swept_curve
                    IF schema_name + 'SWEPT_SURFACE' IN TYPEOF(sf) THEN
                        RETURN(gbsf_check_curve(sf.swept_curve, schema_name));
                    END_IF;
                    END_IF;
                    END_IF;
                    END_IF;
                    ENDIF;
                ENDIF;
                RETURN(FALSE);

            END_FUNCTION;

            FUNCTION get_basis_surface(
                c: curve_on_surface
            ): surface;
            CASE TYPEOF(c) OF
                'MECHANICAL_DESIGN_SURFACE_SCHEMA.PCURVE' : RETURN
                (c.basis_surface);
                'MECHANICAL_DESIGN_SURFACE_SCHEMA.SURFACE_CURVE' : RETURN
                (c.basis_surface);
                'MECHANICAL_DESIGN_SURFACE_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' : RETURN
                (c.basis_surface);
            END_CASE;
        ENDIF;
    ENDIF;

```

```

END_FUNCTION; -- get_basis_surface

FUNCTION item_in_context(
    item: representation_item;
    ctxt: representation_context
): BOOLEAN;

LOCAL
    i : INTEGER;
    y : SET OF representation_item;
END_LOCAL;
IF SIZEOF(USEDIN(item,
    'MECHANICAL_DESIGN_SURFACE_SCHEMA.REPRESENTATION.ITEMS') * ctxt.
    contexted_representations) > 0 THEN
    RETURN(TRUE);
ELSE
    IF SIZEOF(USEDIN(item,'MECHANICAL_DESIGN_SURFACE_SCHEMA.DEFINITIONAL REPRESENTATION_ITEM
        * ctxt.contexted_definitional_representation_items) > 0 THEN
        RETURN(TRUE);
    ELSE
        y := QUERY ( z <* USEDIN(item,'') | (
            'MECHANICAL_DESIGN_SURFACE_SCHEMA.REPRESENTATION_ITEM' IN
            TYPEOF(z)) );
        IF SIZEOF(y) > 0 THEN
            REPEAT i := 1 TO HIINDEX(y) BY 1;
                IF item_in_context(y[i],ctxt) THEN
                    RETURN(TRUE);
                END_IF;
            END_REPEAT;
            END_IF;
        END_IF;
        RETURN(FALSE);
    END_IF;
END_FUNCTION; -- item_in_context

FUNCTION list_face_loops(
    f: face
): LIST [0:?] OF loop;

LOCAL
    loops : LIST [0:?] OF loop := [];
END_LOCAL;
REPEAT i := 1 TO SIZEOF(f.bounds) BY 1;
    loops := loops + f.bounds[i].bound;

```

```

END_REPEAT;
RETURN(loops);

END_FUNCTION; -- list_face_loops

FUNCTION list_of_topology_reversed(
    a_list: list_of_reversible_topology_item
): list_of_reversible_topology_item;

LOCAL
    the_reverse : list_of_reversible_topology_item;
END_LOCAL;
the_reverse := [];
REPEAT i := 1 TO SIZEOF(a_list) BY 1;
    the_reverse := the_reverse + topology_reversed(a_list[i]);
END_REPEAT;
RETURN(the_reverse);

END_FUNCTION; -- list_of_topology_reversed

FUNCTION list_to_array(
    lis: LIST [0:?] OF GENERIC:t;
    low, u: INTEGER
): ARRAY [low:u] OF GENERIC:t;

LOCAL
    n : INTEGER;
    res : ARRAY [low:u] OF GENERIC:t;
END_LOCAL;
n := SIZEOF(lis);
IF n <> ((u - low) + 1) THEN
    RETURN(?);
ELSE
    REPEAT i := 1 TO n BY 1;
        res[(low + i) - 1] := lis[i];
    END_REPEAT;
    RETURN(res);
END_IF;

END_FUNCTION; -- list_to_array

FUNCTION list_to_set(
    l: LIST [0:?] OF GENERIC:t
): SET OF GENERIC:t;

LOCAL

```

```

        s : SET OF GENERIC:t := [];
END_LOCAL;
REPEAT i := 1 TO SIZEOF(l) BY 1;
  s := s + l[i];
END_REPEAT;
RETURN(s);

END_FUNCTION; -- list_to_set

FUNCTION local_relatives_of_product_definitions(
    definition_set: SET OF product_definition;
    total_definitions: SET OF product_definition;
    relation_subtype: STRING
): SET OF product_definition;

LOCAL
  i : INTEGER;
  local_def : SET OF product_definition := [];
  local_pdr : SET OF product_definition_relationship := [];
  local_total : SET OF product_definition := [];
END_LOCAL;
REPEAT i := 1 TO HIINDEX(definition_set) BY 1;
  local_pdr := local_pdr + USEDIN(definition_set[i],relation_subtype +
    '.RELATING_PRODUCT_DEFINITION');
END_REPEAT;
REPEAT i := 1 TO HIINDEX(local_pdr) BY 1;
  local_def := local_def + local_pdr[i].related_product_definition;
END_REPEAT;
IF (SIZEOF(local_def) - SIZEOF(total_definitions)) = 0 THEN
  RETURN(local_def);
ELSE
  local_total := total_definitions + local_def;
  RETURN(local_def + local_relatives_of_product_definitions(local_def -
    total_definitions,local_total,relation_subtype));
END_IF;

END_FUNCTION; -- local_relatives_of_product_definitions

FUNCTION local_relatives_of_shape_representations(
    shape_representation_set: SET OF shape_representation;
    total_reps: SET OF shape_representation
): SET OF shape_representation;

LOCAL
  local_shape_rep : SET OF shape_representation := [];
  i : INTEGER;

```

```

local_srr      : SET OF shape_representation_relationship := [];
local_total    : SET OF shape_representation := [];
END_LOCAL;
REPEAT i := 1 TO HIINDEX(shape_representation_set) BY 1;
  local_srr := local_srr + USEDIN(shape_representation_set[i],
    'MECHANICAL DESIGN SURFACE SCHEMA.' +
    'SHAPE REPRESENTATION RELATIONSHIP.REP_1');
END_REPEAT;
REPEAT i := 1 TO HIINDEX(local_srr) BY 1;
  local_shape_rep := local_shape_rep + local_srr[i].rep_2;
END_REPEAT;
IF SIZEOF(local_shape_rep - total_reps) = 0 THEN
  RETURN(shape_representation_set);
ELSE
  local_total := total_reps + local_shape_rep;
  RETURN(local_shape_rep + local_relatives_of_shape_representations(
    local_shape_rep - total_reps,local_total));
END_IF;

END_FUNCTION; -- local_relatives_of_shape_representations

FUNCTION make_array_of_array(
  lis: LIST [1:?] OF LIST [1:?] OF GENERIC:t;
  low1, u1, low2, u2: INTEGER
): ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC:t;

LOCAL
  n2  : INTEGER;
  n1  : INTEGER;
  res : ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC:t;
  resl : LIST [1:?] OF ARRAY [low2:u2] OF GENERIC:t;
END_LOCAL;
n1 := SIZEOF(lis);
n2 := SIZEOF(lis[1]);
IF (n1 <> ((u1 - low1) + 1)) AND (n2 <> ((u2 - low2) + 1)) THEN
  RETURN(?);
END_IF;
REPEAT i := 1 TO n1 BY 1;
  IF SIZEOF(lis[i]) <> n2 THEN
    RETURN(?);
  END_IF;
END_REPEAT;
REPEAT i := 1 TO n1 BY 1;
  resl[i] := list_to_array(lis[i],low2,u2);
END_REPEAT;
res := list_to_array(resl,low1,u1);

```

```

RETURN(res);

END_FUNCTION; -- make_array_of_array

FUNCTION masf_curve_check (cv : curve; schema_name : STRING) : BOOLEAN;

-- let those types pass that do not have references to curves
IF SIZEOF ([schema_name + '.B_SPLINE_CURVE', schema_name + '.CONIC',
schema_name + '.LINE', schema_name + '.POLYLINE'] * TYPEOF(cv)) = 1 THEN
    RETURN(TRUE);
ELSE

    -- a curve_replica may reference any curve type of this AIC
    IF schema_name + '.CURVE_REPLICA' IN TYPEOF(cv) THEN
        RETURN(mASF_curve_check(cv.parent_curve, schema_name));
    ELSE

        -- an offset_curve_3d may reference any curve type of this AIC
        IF schema_name + '.OFFSET_CURVE_3D' IN TYPEOF(cv) THEN
            RETURN(mASF_curve_check(cv.basis_curve, schema_name));
        ELSE

            -- for the purpose of this rule a pcurve may reference any curve
            -- that is in the scope of this AIC; ISO 10303-42 implies further
            -- restrictions
            IF schema_name + '.PCURVE' IN TYPEOF(cv) THEN
                RETURN(mASF_curve_check(cv.reference_to_curve.items[1], schema_name));
            ELSE

                -- for the purpose of this rule a surface_curve may reference any curve
                -- that is in the scope of this AIC; ISO 10303-42 implies further
                -- restrictions
                IF schema_name + '.SURFACE_CURVE' IN TYPEOF(cv) THEN
                    RETURN(mASF_curve_check(cv.curve_3d, schema_name));
                END_IF;
                END_IF;
            END_IF;
        END_IF;
    END_IF;
END_IF;

RETURN(FALSE);
END_FUNCTION;

FUNCTION masf_surface_check (surf : surface; schema_name : STRING) : BOOLEAN;

-- let those surface types pass that do not have references to surfaces

```

```

IF SIZEOF ([schema_name + '.B_SPLINE_SURFACE',
            schema_name + '.ELEMENTARY_SURFACE', schema_name + '.SWEPT_SURFACE']
           * TYPEOF(surf)) = 1 THEN
    RETURN(TRUE);
ELSE

    -- an offset_surface may reference any surface type of this AIC
    IF schema_name + '.OFFSET_SURFACE' IN TYPEOF(surf) THEN
        RETURN(mASF_surface_check(surf.basis_surface, schema_name));
    ELSE

        -- a surface_replica may reference any surface type of this AIC
        IF schema_name + '.SURFACE_REPLICA' IN TYPEOF(surf) THEN
            RETURN(mASF_surface_check(surf.parent_surface, schema_name));
        END_IF;
        END_IF;
    END_IF;

    RETURN(FALSE);
END_FUNCTION;

FUNCTION mdp_get_text_literal_mapped_item_in_annotation_text
    (at : annotation_text; schema_name : STRING) :
    BAG [0:?] OF text_literal_mapped_item;

LOCAL
    txlits: BAG OF text_literal_mapped_item := [];
    tsr   : text_string_representation :=
        at\mapped_item.mapping_source.mapped_representation;
END_LOCAL;

-- check whether the input has the right type
IF NOT (schema_name + '.ANNOTATION_TEXT' IN TYPEOF (at)) THEN
    RETURN(txlits);
END_IF;

-- do for each member in the strings of the text_string_representation
REPEAT i := 1 TO SIZEOF (tsr.strings);

    -- if the member is annotation_text, go into recursion
    IF schema_name + '.ANNOTATION_TEXT' IN TYPEOF (tsr.strings[i]) THEN
        txlits := txlits + mdp_get_text_literal_mapped_item_in_annotation_text
            (tsr.strings[i], schema_name);
    ELSE

        -- if the member is text_literal_mapped_item, collect it and go to next

```

```

-- member
IF schema_name + '.TEXT_LITERAL_MAPPED_ITEM' IN TYPEOF
(tsr.strings[i]) THEN
    txlits := txlits + 1;
ELSE

    -- if anything is wrong, return an empty bag
    txlits := [];
    RETURN(txlits);
END_IF;
END_IF;
END_REPEAT;
RETURN(txlits);

END_FUNCTION; -- mdp_get_text_literal_mapped_item_in_annotation_text

FUNCTION mixed_loop_type_set(
    l: SET [0:?] OF loop
): BOOLEAN;

LOCAL
    i : INTEGER;
    poly_loop_type : BOOLEAN;
END_LOCAL;
IF SIZEOF(l) <= 1 THEN
    RETURN(FALSE);
END_IF;
poly_loop_type := 'MECHANICAL_DESIGN_SURFACE_SCHEMA.POLY_LOOP' IN
    TYPEOF(l[1]);
REPEAT i := 2 TO SIZEOF(l) BY 1;
    IF ('MECHANICAL_DESIGN_SURFACE_SCHEMA.POLY_LOOP' IN TYPEOF(l[i]))
        :<>: poly_loop_type THEN
        RETURN(TRUE);
    END_IF;
END_REPEAT;
RETURN(FALSE);

END_FUNCTION; -- mixed_loop_type_set

FUNCTION nmsf_curve_check (cv : curve; schema_name : STRING) : BOOLEAN;

-- let those types pass that do not have references to curves
-- and that are legal curve types in the context of this AIC
IF SIZEOF ([schema_name + '.B_SPLINE_CURVE', schema_name + '.CONIC',
    schema_name + '.LINE', schema_name + '.POLYLINE'] * TYPEOF(cv)) = 1 THEN
    RETURN(TRUE);

```

```

ELSE

-- a curve_replica may reference any curve type of this AIC
IF schema_name + '.CURVE_REPLICA' IN TYPEOF(cv) THEN
  RETURN(nmsf_curve_check(cv.parent_curve, schema_name));
ELSE

-- an offset_curve_3d may reference any curve type of this AIC
IF schema_name + '.OFFSET_CURVE_3D' IN TYPEOF(cv) THEN
  RETURN(nmsf_curve_check(cv.basis_curve, schema_name));
ELSE

-- for the purpose of this rule a pcurve may reference any curve
-- that is in the scope of this AIC; ISO 10303-42 implies further
-- restrictions
IF schema_name + '.PCURVE' IN TYPEOF(cv) THEN
  RETURN(nmsf_curve_check(cv.reference_to_curve.items[1], schema_name));
ELSE

-- for the purpose of this rule a surface_curve may reference any curve
-- that is in the scope of this AIC; ISO 10303-42 implies further
-- restrictions
  IF schema_name + '.SURFACE_CURVE' IN TYPEOF(cv) THEN
    RETURN(nmsf_curve_check(cv.curve_3d, schema_name));
  END_IF;
  END_IF;
END_IF;

RETURN(FALSE);
END_FUNCTION;

FUNCTION nmsf_surface_check (surf : surface; schema_name : STRING) : BOOLEAN;

-- let those surface types pass that do not have references to surfaces
-- and that are legal surface types in the context of this AIC
IF SIZEOF ([schema_name + '.B_SPLINE_SURFACE',
schema_name + '.ELEMENTARY_SURFACE', schema_name + '.SWEPT_SURFACE']
* TYPEOF(surf)) = 1 THEN
  RETURN(TRUE);
ELSE

-- an offset_surface may reference any surface type of this AIC
IF schema_name + '.OFFSET_SURFACE' IN TYPEOF(surf) THEN
  RETURN(nmsf_surface_check(surf.basis_surface, schema_name));

```

```

ELSE

-- a surface_replica may reference any surface type of this AIC
IF schema_name + '.SURFACE_REPLICA' IN TYPEOF(surf) THEN
    RETURN(nmsf_surface_check(surf.parent_surface, schema_name));
END_IF;
END_IF;
END_IF;

RETURN(FALSE);
END_FUNCTION;

FUNCTION normalise(
    arg: vector_or_direction
): vector_or_direction;

LOCAL
    ndim    : INTEGER;
    v       : direction;
    vec     : vector;
    mag     : REAL;
    result  : vector_or_direction;
END_LOCAL;
IF NOT EXISTS(arg) THEN
    result := ?>;
ELSE
    ndim := arg.dim;
    IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
        BEGIN
            vec := arg;
            v := arg.orientation;
            IF arg.magnitude = 0 THEN
                RETURN(?);
            ELSE
                vec.magnitude := 1;
            END_IF;
        END;
    ELSE
        v := arg;
    END_IF;
    mag := 0;
    REPEAT i := 1 TO ndim BY 1;
        mag := mag + (v.direction_ratios[i] * v.direction_ratios[i]);
    END_REPEAT;
    IF mag > 0 THEN
        mag := SQRT(mag);
    END_IF;
END_IF;

```

```

    REPEAT i := 1 TO ndim BY 1;
        v.direction_ratios[i] := v.direction_ratios[i] / mag;
    END_REPEAT;
    IF 'MECHANICAL DESIGN_SURFACE_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
        vec.orientation := v;
        result := vec;
    ELSE
        result := v;
    END_IF;
    ELSE
        RETURN(?);
    END_IF;
    RETURN(result);
END_IF;

END_FUNCTION; -- normalise

FUNCTION orthogonal_complement(
    vec: direction
): direction;

LOCAL
    result : direction;
END_LOCAL;
IF (vec.dim <> 2) OR (NOT EXISTS(vec)) THEN
    RETURN(?);
ELSE
    result.direction_ratios[1] := -vec.direction_ratios[2];
    result.direction_ratios[2] := vec.direction_ratios[1];
    RETURN(result);
END_IF;

END_FUNCTION; -- orthogonal_complement

FUNCTION path_head_to_tail(
    a_path: path
): BOOLEAN;

LOCAL
    n : INTEGER;
    p : BOOLEAN := TRUE;
END_LOCAL;
n := SIZEOF(a_path.edge_list);
REPEAT i := 2 TO n BY 1;
    p := p AND (a_path.edge_list[i - 1].edge_end :: a_path.edge_list[i]
                .edge_start);

```

```

END_REPEAT;
RETURN(p);

END_FUNCTION; -- path_head_to_tail

FUNCTION path_reversed(
    a_path: oriented_path
): path;

LOCAL
    the_reverse : path;
END_LOCAL;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.ORIENTED_PATH' IN TYPEOF(a_path)
    THEN
        the_reverse := oriented_path(a_path.path_element,NOT a_path.
            orientation);
    ELSE
        the_reverse := oriented_path(a_path,FALSE);
    END_IF;
RETURN(the_reverse);

END_FUNCTION; -- path_reversed

FUNCTION relationships_are_coordinated(
    sdr: SET OF shape_definition_representation
): BOOLEAN;

LOCAL
    sr1 : shape_representation;
    sr2 : shape_representation;
    i   : INTEGER;
    j   : INTEGER;
    pd1 : product_definition;
    pd2 : product_definition;
END_LOCAL;
REPEAT i := 1 TO HIINDEX(sdr) BY 1;
    REPEAT j := 1 TO HIINDEX(sdr) BY 1;
        IF i <> j THEN
            sr1 := sdr[i].representation_model;
            sr2 := sdr[j].representation_model;
            pd1 := represented_product_definition(sdr[i]);
            pd2 := represented_product_definition(sdr[j]);
            IF (sr2 IN relatives_of_shape_representations([sr1])) XOR (pd2
                IN relatives_of_product_definitions([pd1],'MECHANICAL_DESIGN_SURFACE_SCHEMA.PROD
                THEN
                    RETURN(FALSE);

```

```

        END_IF;
    END_IF;
END_REPEAT;
END_REPEAT;
RETURN(TRUE);

END_FUNCTION; -- relationships_are_coordinated

FUNCTION relatives_of_product_definitions(
    definition_set: SET OF product_definition;
    relation_subtype: STRING
): SET OF product_definition;

FUNCTION local_relatives_of_product_definitions(
    definition_set: SET OF product_definition;
    total_definitions: SET OF product_definition;
    relation_subtype: STRING
): SET OF product_definition;

LOCAL
    i : INTEGER;
    local_def : SET OF product_definition := [];
    local_pdr : SET OF product_definition_relationship := [];
    local_total : SET OF product_definition := [];
END_LOCAL;
REPEAT i := 1 TO HIIINDEX(definition_set) BY 1;
    local_pdr := local_pdr + USEDIN(definition_set[i],relation_subtype
        + '.RELATING_PRODUCT_DEFINITION');
END_REPEAT;
REPEAT i := 1 TO HIIINDEX(local_pdr) BY 1;
    local_def := local_def + local_pdr[i].related_product_definition;
END_REPEAT;
IF (SIZEOF(local_def) - SIZEOF(total_definitions)) = 0 THEN
    RETURN(local_def);
ELSE
    local_total := total_definitions + local_def;
    RETURN(local_def + local_relatives_of_product_definitions(
        local_def - total_definitions,local_total,relation_subtype));
END_IF;

END_FUNCTION; -- local_relatives_of_product_definitions
RETURN(local_relatives_of_product_definitions(definition_set,
    definition_set,relation_subtype));

END_FUNCTION; -- relatives_of_product_definitions

```

```

FUNCTION relatives_of_shape_representations(
    shape_representation_set: SET OF shape_representation
): SET OF shape_representation;

FUNCTION local_relatives_of_shape_representations(
    shape_representation_set: SET OF shape_representation;
    total_reps: SET OF shape_representation
): SET OF shape_representation;

LOCAL
    local_shape_rep : SET OF shape_representation := [];
    i              : INTEGER;
    local_srr      : SET OF shape_representation_relationship := [];
    local_total    : SET OF shape_representation := [];
END_LOCAL;

REPEAT i := 1 TO HIINDEX(shape_representation_set) BY 1;
    local_srr := local_srr + USEDIN(shape_representation_set[i],
        'MECHANICAL DESIGN SURFACE SCHEMA.' +
        'SHAPE REPRESENTATION RELATIONSHIP.REP_1');
END_REPEAT;

REPEAT i := 1 TO HIINDEX(local_srr) BY 1;
    local_shape_rep := local_shape_rep + local_srr[i].rep_2;
END_REPEAT;

IF SIZEOF(local_shape_rep - total_reps) = 0 THEN
    RETURN(shape_representation_set);
ELSE
    local_total := total_reps + local_shape_rep;
    RETURN(local_shape_rep + local_relatives_of_shape_representations(
        local_shape_rep - total_reps,local_total));
END_IF;

END_FUNCTION; -- local_relatives_of_shape_representations
RETURN(local_relatives_of_shape_representations(
    shape_representation_set,shape_representation_set));

END_FUNCTION; -- relatives_of_shape_representations

FUNCTION represented_product_definition(
    sdr: shape_definition_representation
): product_definition;

LOCAL
    prod_def_shape_string : STRING := 'PRODUCT_PROPERTY_DEFINITION' +
        '_SCHEMA.PRODUCT_DEFINITION_SHAPE';
    prod_def_rel_string   : STRING :=
        'MECHANICAL DESIGN SURFACE SCHEMA.' +

```

```

        'PRODUCT_DEFINITION_RELATIONSHIP';

END_LOCAL;
IF SIZEOF(TYPEOF(sdr.representation_of) * [prod_def_shape_string]) = 0
    THEN
    IF SIZEOF(TYPEOF(sdr.representation_of.shape.definition) * [
        prod_def_rel_string]) = 0 THEN
        RETURN(sdr.representation_of.shape.definition);
    ELSE
        RETURN(sdr.representation_of.shape.definition.
            related_product_definition);
    END_IF;
ELSE
    IF SIZEOF(TYPEOF(sdr.representation_of.definition) * [
        prod_def_rel_string]) = 0 THEN
        RETURN(sdr.representation_of.definition);
    ELSE
        RETURN(sdr.representation_of.definition.related_product_definition);
    END_IF;
END_IF;

END_FUNCTION; -- represented_product_definition

FUNCTION scalar_times_vector(
    scalar: NUMBER;
    vec: vector_or_direction
): vector;

LOCAL
    v      : direction;
    mag   : REAL;
    result : vector;
END_LOCAL;
IF (NOT EXISTS(scalar)) OR (NOT EXISTS(vec)) THEN
    result := ?;
ELSE
    IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.VECTOR' IN TYPEOF(vec) THEN
        v := vec.orientation;
        mag := scalar * vec.magnitude;
    ELSE
        v := vec;
        mag := scalar;
    END_IF;
    result.orientation := normalise(v);
    result.magnitude := mag;
    RETURN(result);
END_IF;

```

```

END_FUNCTION; -- scalar_times_vector

FUNCTION second_proj_axis(
    z_axis, x_axis, arg: direction
): direction;

LOCAL
    temp    : direction;
    v       : direction;
    y_axis  : vector;
END_LOCAL;
IF NOT EXISTS(arg) THEN
    v := direction([0,1,0]);
ELSE
    v := arg;
END_IF;
temp := scalar_times_vector(dot_product(v,z_axis),z_axis);
y_axis := vector_diff(v,temp);
temp := scalar_times_vector(dot_product(v,x_axis),x_axis);
y_axis := vector_diff(y_axis,temp);
y_axis := normalise(y_axis);
RETURN(y_axis.orientation);

END_FUNCTION; -- second_proj_axis

FUNCTION set_of_topology_reversed(
    a_set: set_of_reversible_topology_item
): set_of_reversible_topology_item;

LOCAL
    the_reverse : set_of_reversible_topology_item;
END_LOCAL;
the_reverse := [];
REPEAT i := 1 TO SIZEOF(a_set) BY 1;
    the_reverse := the_reverse + topology_reversed(a_set[i]);
END_REPEAT;
RETURN(the_reverse);

END_FUNCTION; -- set_of_topology_reversed

FUNCTION shell_reversed(
    a_shell: shell
): shell;

LOCAL

```

```

        the_reverse : shell;
END_LOCAL;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.ORIENTED_OPEN_SHELL' IN TYPEOF(
    a_shell) THEN
    the_reverse := oriented_open_shell(a_shell.open_shell_element,NOT
        a_shell.orientation);
ELSE
    IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.OPEN_SHELL' IN TYPEOF(a_shell)
        THEN
            the_reverse := oriented_open_shell(a_shell, FALSE);
    ELSE
        IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.ORIENTED_CLOSED_SHELL' IN
            TYPEOF(a_shell) THEN
            the_reverse := oriented_closed_shell(a_shell.
                closed_shell_element, FALSE);
        ELSE
            IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.CLOSED_SHELL' IN TYPEOF(
                a_shell) THEN
                the_reverse := oriented_closed_shell(a_shell, FALSE);
            ELSE
                the_reverse := ?;
            END_IF;
        END_IF;
    END_IF;
RETURN(the_reverse);

END_FUNCTION; -- shell_reversed

FUNCTION surface_weights_positive(
    b: rational_b_spline_surface
): BOOLEAN;

LOCAL
    result : BOOLEAN := TRUE;
END_LOCAL;
REPEAT i := 0 TO b.u_upper BY 1;
    REPEAT j := 0 TO b.v_upper BY 1;
        IF b.weights[i][j] <= 0 THEN
            result := FALSE;
            RETURN(result);
        END_IF;
    END_REPEAT;
END_REPEAT;
RETURN(result);

```

```

END_FUNCTION; -- surface_weights_positive

FUNCTION topology_reversed(
    an_item: reversible_topology
): reversible_topology;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.EDGE' IN TYPEOF(an_item) THEN
    RETURN(edge_reversed(an_item));
END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.PATH' IN TYPEOF(an_item) THEN
    RETURN(path_reversed(an_item));
END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.FACE_BOUND' IN TYPEOF(an_item)
    THEN
    RETURN(face_bound_reversed(an_item));
END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.FACE' IN TYPEOF(an_item) THEN
    RETURN(face_reversed(an_item));
END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.SHELL' IN TYPEOF(an_item) THEN
    RETURN(shell_reversed(an_item));
END_IF;
IF 'SET' IN TYPEOF(an_item) THEN
    RETURN(set_of_topology_reversed(an_item));
END_IF;
IF 'LIST' IN TYPEOF(an_item) THEN
    RETURN(list_of_topology_reversed(an_item));
END_IF;
RETURN(?);

END_FUNCTION; -- topology_reversed

FUNCTION using_definitional_representation_items(
    item: representation_item
): SET OF definitional_representation_item;

LOCAL
    results          : SET OF definitional_representation_item;
    i                : INTEGER;
    intermediate_items : SET OF representation_item;
END_LOCAL;
results := USEDIN(item,'MECHANICAL_DESIGN_SURFACE_SCHEMA.DEFINITIONAL_REPRESENTATION_ITEM');
intermediate_items := QUERY ( z <* USEDIN(item,'') | (
    'MECHANICAL_DESIGN_SURFACE_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z)) );
IF SIZEOF(intermediate_items) > 0 THEN
    REPEAT i := 1 TO HINDEX(intermediate_items) BY 1;
        results := results + using_definitional_representation_items(

```

```

        intermediate_items[i]);
    END_REPEAT;
END_IF;
RETURN(results);

END_FUNCTION; -- using_definitional_representation_items

FUNCTION using_representations(
    item: representation_item
): SET OF representation;

LOCAL
    results          : SET OF representation;
    i                : INTEGER;
    intermediate_items : SET OF representation_item;
END_LOCAL;
results := USEDIN(item,
    'MECHANICAL_DESIGN_SURFACE_SCHEMA.REPRESENTATION.ITEMS');
intermediate_items := QUERY ( z <* USEDIN(item,'') | (
    'MECHANICAL_DESIGN_SURFACE_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z)) );
IF SIZEOF(intermediate_items) > 0 THEN
    REPEAT i := 1 TO HINDEX(intermediate_items) BY 1;
        results := results + using_representations(intermediate_items[i]);
    END_REPEAT;
END_IF;
RETURN(results);

END_FUNCTION; -- using_representations

FUNCTION valid_units(
    m: measure_with_unit
): BOOLEAN;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.LENGTH_MEASURE' IN TYPEOF(m.
    value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,1,0,0,0,0,0) THEN
            RETURN(FALSE);
        END_IF;
    END_IF;
    IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.TIME_MEASURE' IN TYPEOF(m.
        value_component) THEN
        IF derive_dimensional_exponents(m.unit_component) <>
            dimensional_exponents(0,0,1,0,0,0,0) THEN
                RETURN(FALSE);
            END_IF;
    END_IF;

```

```
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.ELECTRIC_CURRENT_MEASURE' IN
    TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,1,0,0,0) THEN
            RETURN(FALSE);
        END_IF;
    END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.THERMODYNAMIC_TEMPERATURE_MEASURE'
    IN TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,1,0,0) THEN
            RETURN(FALSE);
        END_IF;
    END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.AMOUNT_OF_SUBSTANCE_MEASURE' IN
    TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,1,0) THEN
            RETURN(FALSE);
        END_IF;
    END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.LUMINOUS_INTENSITY_MEASURE' IN
    TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,0,1) THEN
            RETURN(FALSE);
        END_IF;
    END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.PLANE_ANGLE_MEASURE' IN TYPEOF(m.
    value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,0,0) THEN
            RETURN(FALSE);
        END_IF;
    END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.SOLID_ANGLE_MEASURE' IN TYPEOF(m.
    value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,0,0) THEN
            RETURN(FALSE);
        END_IF;
    END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.AREA_MEASURE' IN TYPEOF(m.
    value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(2,0,0,0,0,0,0) THEN
```

```

        RETURN(FALSE);
    END_IF;
END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.VOLUME_MEASURE' IN TYPEOF(m.
    value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(3,0,0,0,0,0,0) THEN
            RETURN(FALSE);
    END_IF;
END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.RATIO_MEASURE' IN TYPEOF(m.
    value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,0,0) THEN
            RETURN(FALSE);
    END_IF;
END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.POSITIVE_LENGTH_MEASURE' IN
    TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(1,0,0,0,0,0,0) THEN
            RETURN(FALSE);
    END_IF;
END_IF;
IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.POSITIVE_PLANE_ANGLE_MEASURE' IN
    TYPEOF(m.value_component) THEN
    IF derive_dimensional_exponents(m.unit_component) <>
        dimensional_exponents(0,0,0,0,0,0,0) THEN
            RETURN(FALSE);
    END_IF;
END_IF;
RETURN(TRUE);

END_FUNCTION; -- valid_units

FUNCTION vector_diff(
    arg1, arg2: vector_or_direction
): vector;

LOCAL
    ndim    : INTEGER;
    mag2    : REAL;
    mag1    : REAL;
    mag     : REAL;
    res     : direction;
    vec1   : direction;

```

```

    vec2    : direction;
    result : vector;
END_LOCAL;
IF ((arg1.dim <> arg2.dim) OR (NOT EXISTS(arg1))) OR (NOT EXISTS(arg2))
    THEN
    result := ?;
ELSE
BEGIN
    IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
        mag1 := arg1.magnitude;
        vec1 := arg1.orientation;
    ELSE
        mag1 := 1;
        vec1 := arg1;
    END_IF;
    IF 'MECHANICAL_DESIGN_SURFACE_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN
        mag2 := arg2.magnitude;
        vec2 := arg2.orientation;
    ELSE
        mag2 := 1;
        vec2 := arg2;
    END_IF;
    vec1 := normalise(vec1);
    vec2 := normalise(vec2);
    ndim := SIZEOF(vec1.direction_ratios);
    REPEAT i := 1 TO ndim BY 1;
        res.direction_ratios[i] := (mag1 * vec1.direction_ratios[i]) - (
            mag2 * vec2.direction_ratios[i]);
        mag := mag + (res.direction_ratios[i] * res.direction_ratios[i]);
    END_REPEAT;
    result.magnitude := SQRT(mag);
    result.orientation := res;
END;
RETURN(result);
END_IF;

END_FUNCTION; -- vector_diff

END_SCHEMA; -- mechanical_design_surface_schema
(*

```

**Annex B**  
(normative)

**AIM short names**

Table B.1 provides the short names of entities specified in the AIM of this part of ISO 10303. Requirements on the use of the short names are found in the implementation methods included in ISO 10303.

**Table B.1 – Entities and their abbreviations for Mechanical Design Using Surface Representation application protocol**

AIM entity	Abbreviation
ADVANCED_FACE	ADVFACT
ANNOTATION_OCCURRENCE	ANNOCC
ANNOTATION_TEXT	ANNTXT
ANNOTATION_TEXT_MAP	ANTXMP
ANNOTATION_TEXT_OCCURRENCE	ANTXOC
APPLICATION_CONTEXT	APPCNT
APPLICATION_CONTEXT_ELEMENT	APCNEL
ASSEMBLY_COMPONENT_USAGE	ASCMUS
AXIS1_PLACEMENT	AX1PLC
AXIS2_PLACEMENT_2D	A2PL2D
AXIS2_PLACEMENT_3D	A2PL3D
BACKGROUND_COLOUR	BCKCLR
BEZIER_CURVE	BZRCRV
BEZIER_SURFACE	BZRSRF
BOUNDARY_CURVE	BNDCRV
BOUNDED_CURVE	BNDCR
BOUNDED_SURFACE	BNDSRF
B_SPLINE_CURVE	BSPCR
B_SPLINE_CURVE_WITH_KNOTS	BSCWK
B_SPLINE_SURFACE	BSPSR
B_SPLINE_SURFACE_WITH_KNOTS	BSSWK
CAMERA_IMAGE	CMRIMG
CAMERA_MODEL	CMRMDL
CAMERA_MODEL_D3	CMMDD3
CAMERA_USAGE	CMRUSG
CARTESIAN_POINT	CRTPNT
CARTESIAN_TRANSFORMATION_OPERATOR	CRTROP
CARTESIAN_TRANSFORMATION_OPERATOR_3D	CTO3
CIRCLE	CIRCLE
CLOSED_SHELL	CLSSH
COLOUR	COLOUR
COLOUR_RGB	CLRRGB
COLOUR_SPECIFICATION	CLRSPEC
COMPOSITE_CURVE	CMPCRV
COMPOSITE_CURVE_ON_SURFACE	CCOS
COMPOSITE_CURVE_SEGMENT	CMCRSG

Table B.1 (continued)

AIM entity	Abbreviation
CONIC	CONIC
CONICAL_SURFACE	CNCSRF
CONNECTED_FACE_SET	CNFNST
CONVERSION_BASED_UNIT	CNBSUN
CURVE	CURVE
CURVE_BOUNDED_SURFACE	CRBNSR
CURVE_REPLICA	CRVRPL
CURVE_STYLE	CRVSTY
CURVE_STYLE_FONT	CRSTFN
CURVE_STYLE_FONT_PATTERN	CSFP
CURVE_STYLE_RENDERING	CRSTRN
CYLINDRICAL_SURFACE	CYLSRF
DEFINITIONAL REPRESENTATION_ITEM	DFRPIT
DEGENERATE_PCURVE	DGNPCR
DESIGN_CONTEXT	DSGCNT
DIMENSIONAL_EXPONENTS	DMNEXP
DIRECTION	DRCTN
EDGE	EDGE
EDGE_CURVE	EDGCRV
EDGE_LOOP	EDGLP
ELEMENTARY_SURFACE	ELMSRF
ELLIPSE	ELLP
EVALUATED_DEGENERATE_PCURVE	EVDGPC
FACE	FACE
FACE_BASED_SURFACE_MODEL	FBSM
FACE_BOUND	FCBND
FACE_OUTER_BOUND	FCOTBN
FACE_SURFACE	FCSR
FUNCTIONALLY_DEFINED_TRANSFORMATION	FNDFT
GEOMETRIC_REPRESENTATION_CONTEXT	GMRPCN
GEOMETRIC_REPRESENTATION_ITEM	GMRPIT
GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION	GBSSR
GEOMETRIC_SET	GMTST
GEOMETRIC_SET_REPLICA	GMSTRP
GLOBAL_UNIT_ASSIGNED_CONTEXT	GUAC
GROUP	GROUP
GROUP_ASSIGNMENT	GRPASS
HYPERBOLA	HYPRBL
INTERSECTION_CURVE	INTCRV

Table B.1 (continued)

AIM entity	Abbreviation
LENGTH_UNIT	LNGUNT
LINE	LINE
LOOP	LOOP
MANIFOLD_SURFACE_SHAPE_REPRESENTATION	MSSR
MAPPED_ITEM	MPPITM
MEASURE_WITH_UNIT	MSWTUN
MECHANICAL_CONTEXT	MCHCNT
MECHANICAL_DESIGN_GROUP_ASSIGNMENT	MDGA
MECHANICAL_DESIGN_NAME_ASSIGNMENT	MDNA
MECHANICAL_DESIGN_PRESENTATION_AREA	MDPA
MECHANICAL_DESIGN_PRESENTATION_REPRESENTATION	MDPR
MECHANICAL_DESIGN_PRE_DEFINED_TEXT_FONT	MDPDTF
MECHANICAL_DESIGN_PRE_DEFINED_TEXT_STYLE	MDPDTS
NAMED_UNIT	NMDUNT
NAME_ASSIGNMENT	NMASS
NON_MANIFOLD_SURFACE_SHAPE_REPRESENTATION	NMSSR
OFFSET_CURVE_3D	OFCR3D
OFFSET_SURFACE	OFFSRF
OPEN_SHELL	OPNSHL
ORIENTED_EDGE	ORNEDG
ORIENTED_FACE	ORNFC
OUTER_BOUNDARY_CURVE	OTBNCR
PARABOLA	PRBL
PARAMETRIC_REPRESENTATION_CONTEXT	PRRPCN
PATH	PATH
PCURVE	PCURVE
PLACEMENT	PLCMNT
PLANAR_BOX	PLNBX
PLANAR_EXTENT	PLNEXT
PLANE	PLANE
PLANE_ANGLE_MEASURE_WITH_UNIT	PAMWU
PLANE_ANGLE_UNIT	PLANUN
POINT	POINT
POINT_ON_CURVE	PNONCR
POINT_ON_SURFACE	PNONSR
POINT_STYLE	PNTSTY
POLYLINE	PLYLN
PRESENTATION_AREA	PRSAR
PRESENTATION_LAYER_ASSIGNMENT	PRLYAS
PRESENTATION_LAYER_USAGE	PRLYUS
PRESENTATION_REPRESENTATION	PRSRPR

**Table B.1 (continued)**

<b>AIM entity</b>	<b>Abbreviation</b>
PRESENTATION_SIZE	PRSSZ
PRESENTATION_STYLE_ASSIGNMENT	PRSTAS
PRESENTATION_STYLE_BY_CONTEXT	PSBC
PRESENTATION_VIEW	PRSVW
PRE_DEFINED_ITEM	PRDFIT
PRE_DEFINED_PRESENTATION_STYLE	PDPS
PRE_DEFINED_TEXT_FONT	PDTF
PRODUCT	PRDCT
PRODUCT_CATEGORY	PRDCTG
PRODUCT_CONTEXT	PRDCNT
PRODUCT_DATA REPRESENTATION_VIEW	PDRV
PRODUCT_DATA REPRESENTATION_VIEW_WITH_HLHSR	PDRVWH
PRODUCT_DEFINITION	PRDDFN
PRODUCT_DEFINITION_CONTEXT	PRDFCN
PRODUCT_DEFINITION_RELATIONSHIP	PRDFRL
PRODUCT_DEFINITION_SHAPE	PRDFSH
PRODUCT_DEFINITION_USAGE	PRDFUS
PRODUCT RELATED PRODUCT CATEGORY	PRPC
PRODUCT_VERSION	PRDVRS
QUASI_UNIFORM_CURVE	QSUNCR
QUASI_UNIFORM_SURFACE	QSUNSR
RATIONAL_B_SPLINE_CURVE	RBSC
RATIONAL_B_SPLINE_SURFACE	RBSS
RECTANGULAR_COMPOSITE_SURFACE	RCCMSR
RECTANGULAR_TRIMMED_SURFACE	RCTRSL
REPARAMETRISED_COMPOSITE_CURVE_SEGMENT	RCCS
REPRESENTATION	RPRSNT
REPRESENTATION_CONTEXT	RPRCNT
REPRESENTATION_DEPENDENT_STYLED_ITEM	RDSI
REPRESENTATION_ITEM	RPRITM
REPRESENTATION_ITEM WITHOUT STYLE	RIWS
REPRESENTATION_MAP	RPRMP
REPRESENTATION_RELATIONSHIP	RPRRLT
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION	RRWT
SEAM_CURVE	SMCRV
SHAPE_ASPECT	SHPASP
SHAPE_DEFINITION REPRESENTATION	SHDFRP
SHAPE_PRESENTATION	SHPRPR
SHELL_BASED_SURFACE_MODEL	SBSM
SI_UNIT	SUNT
SPHERICAL_SURFACE	SPHSRF

**Table B.1 (concluded)**

<b>AIM entity</b>	<b>Abbreviation</b>
STYLED_ITEM	STYITM
SURFACE	SRFC
SURFACE_CURVE	SRFCRV
SURFACE_OF_LINEAR_EXTRUSION	SL
SURFACE_OF_REVOLUTION	SROFRV
SURFACE_PATCH	SRFPTC
SURFACE_RENDERING_PROPERTIES	SRRNPR
SURFACE_REPLICA	SRFRPL
SURFACE_SIDE_STYLE	SRSDST
SURFACE_STYLE_BOUNDARY	SRSTBN
SURFACE_STYLE_CONTROL_GRID	SSCG
SURFACE_STYLE_PARAMETER_LINE	SSPL
SURFACE_STYLE_RENDERING	SRSTRN
SURFACE_STYLE_SEGMENTATION_CURVE	SSSC
SURFACE_STYLE_SILHOUETTE	SRSTSL
SURFACE_STYLE_USAGE	SRSTUS
SWEPT_SURFACE	SWPSRF
SYMBOL REPRESENTATION	SYMRPR
TEXT_LITERAL_MAPPED_ITEM	TLM
TEXT_LITERAL_SYMBOL	TXLTSY
TEXT_STRING REPRESENTATION	TXSTRP
TEXT_SYMBOL	TXTSYM
TOPOLOGICAL REPRESENTATION_ITEM	TPRPIT
TOROIDAL_SURFACE	TRDSRF
TRIMMED_CURVE	TRMCRV
UNIFORM_CURVE	UNFCRV
UNIFORM_SURFACE	UNFSRF
VECTOR	VECTOR
VERTEX	VERTEX
VERTEX_LOOP	VRTLP
VERTEX_POINT	VRTPNT
VIEW_DEPENDENT_ANNOTATION_REPRESENTATION	VDAR
VIEW_VOLUME	VWVLM

NOTE – This list of short names is provisional and may be amended when a computer generated list is available.

## Annex C

(normative)

### Protocol Implementation Conformance Statement (PICS) pro-forma

This clause lists the optional elements of this part of ISO 10303. An implementation may choose to support any combination of these optional elements. However, certain combinations of options are likely to be implemented together. These combinations are called conformance classes and are described in the subclauses of this annex.

This annex is in the form of a questionnaire. This questionnaire is intended to be filled out by the implementor and may be used in preparation for conformance testing by a testing laboratory. The completed PICS proforma is referred to as a PICS.

The information in the PICS may be used to identify the appropriate elements of the abstract test suite for use in a conformance assessment test campaign.

Six conformance classes are identified in this part of ISO 10303. A conforming implementation shall support one of these classes. These classes are specified in clause 6 of ISO 10303-205.

Questionnaire:

1. Please provide an identifier for the product or system for which conformance is claimed.

Product name and version identification:

.....

2. Please indicate the implementation method:

ISO 10303-21 Exchange Structure – preprocessor Preprocessor version identification:

.....

ISO 10303-21 Exchange Structure – postprocessor Postprocessor version identification:

.....

ISO 10303-22 Standard Data Access Interface SDAI version identification:

.....

3. Please indicate the classes for which conformance is claimed:

Class 1: geometrically bounded surface shape

Class 2: geometrically bounded surface shape with presentation views

Class 3: manifold surface shape

Class 4: manifold surface shape with presentation views

Class 5: non-manifold surface shape

Class 6: non-manifold surface shape with presentation views

ISO/CD 10303-205

## Annex D (normative)

### Implementation method specific requirements

An implementation method defines the exchange characteristics and representation that are required by this part of ISO 10303. Conformance to this part of ISO 10303 shall be realized using one of the implementation methods specified in ISO 10303.

This annex specifies additional requirements for the use of this part of ISO 10303 with an exchange structure as specified in ISO 10303-21. This annex specifies requirements on the header section of an exchange structure. This annex also specifies rules to achieve the requirement for a stable representation of numbers within a **shape\_representation**.

#### D.1 Reference to AIM schema

The attribute **schema\_identifiers** of the entity **file\_schema** of the header section of an exchange structure shall for the purpose of this part of ISO 10303 include in its list of **schema\_names** the name of this schema, "mechanical\_design\_surface\_schema", and only this name.

#### D.2 Accuracy of numbers

This part utilizes both integer and real numbers. A stable representation of reals is required.

Within the area of mechanical design different categories of tolerances are used: model space tolerance, mechanical tolerance, model tolerance, and numerical tolerance. A tolerance is a distance that has within a given context no impact on the interpretation of a model. A number is considered the same whether the tolerance value is added or not. The following definitions of the tolerance categories above apply:

**D.2.16 model space:** the domain of a surface model. Model coordinates may have values between -10000 and +10000.

**D.2.17 mechanical tolerance:** the tolerance used within the context of mechanical manufacturing. Mechanical tolerances depend on the characteristics of the manufacturing tools.

**D.2.18 model tolerance:** the tolerance used within the context of mechanical design. Model tolerances need to be smaller than mechanical tolerances, but greater than numerical tolerances.

**D.2.19 numerical tolerance:** the tolerance used for digital representation of numbers. Numerical tolerances depend on the kind of representation of a number and on computer characteristics.

Within the context of this part of ISO 10303 two requirements with respect to tolerances are considered important:

- the intervals for the different tolerances mentioned above shall not overlap;

**Table D.1 – Tolerance categories and intervals for this part of ISO 10303**

Tolerance category	Tolerance interval - length	Tolerance interval - angle
Model space	$[-10000, 10000]$	$[0, 360]$
Mechanical tolerance	$[10^{-1}, 10^{-3}]$	$[10^{-1}, 10^{-3}]$
Model tolerance	$[10^{-5}, 10^{-7}]$	$[10^{-5}, 10^{-7}]$
Numerical tolerance	$[10^{-10}, 10^{-12}]$	$[10^{-10}, 10^{-12}]$

- a given number shall meet the actual tolerance that is required by the tolerance category of its context; its accuracy shall be neither lower nor higher.

The motivation for these requirements may be seen in the examples below:

#### EXAMPLES

7 – Depending on the tolerance values applied there may be gaps between faces in solid models. The model tolerance of a modelling system needs to be greater than the actual gaps to make the system evaluate the model as a correct solid. For the purpose of manufacturing the model tolerance must be smaller than the mechanical tolerance. Else a manufacturing tool would discover gaps where the modelling system would not see any.

8 – Numerical calculations introduce and accumulate rounding errors. To avoid that these errors have any influence on a model, these errors, i.e. the numerical tolerance, must be smaller than the model tolerance.

9 – Modelling systems use internally different model tolerances. In one system two points with slightly different coordinate values may be evaluated to be identical. In another system these points will be evaluated to be two different instances. An agreement is required for exchange structures on when these points shall be identical.

To fulfil the two requirements listed above, this part of ISO 10303 deduces from industrial practice intervals for the categories of mechanical and model tolerance. Table ?? provides tolerance intervals for these categories measured in model space units. Different intervals are provided for **length\_measures** and **plane\_angle\_measures**:

Non-overlapping intervals in the given ranges can only be achieved if the numerical tolerance is small. Single precision numbers are not sufficient. This part of ISO 10303 requires the use of double precision numbers for applications that use exchange structures.

As this part of ISO 10303 is for the representation of design models, values are specified for the category of model tolerance. Numbers representing **length\_measures** and **plane\_angle\_measures** shall have the following accuracy measured in model space units:

- for **length\_measures**: minimum:  $10^{-5}$ ; maximum:  $10^{-7}$ .
- for **plane\_angle\_measures**: minimum:  $10^{-5}$ ; maximum:  $10^{-7}$ .

A **shape\_representation** in the context of this part of ISO 10303 shall only contain measures with accuracies as indicated in the list above.

## Annex E (informative)

### Application activity model

The application activity model (AAM) is provided to aid in understanding the scope and information requirements defined in this application protocol. The model is presented as a set of definitions of the activities and the data and a set of activity figures. The AAM covers activities which go beyond the scope of this application protocol.

In scope means:

- for activities: they either contribute to or use information that is in scope of this part of ISO 10303;
- for inputs: they comprise information that is - at least partly - within the scope of this part;
- for controls: they serve to ensure that the output of an activity is in conformance with the scope of this part;
- for outputs: they comprise information that is - at least partly - within the scope of this part.
- for mechanisms: they are used to generate product models the domain of which is in scope of this part.

The diagrams use the IDEF0 activity modelling format.

The viewpoint of the application activity model is that of a designer of mechanical products. Design and engineering activities and their needs for digital data representation are focused upon.

#### E.1 Application activity model definitions

The following terms are used in the application activity model. Terms marked with an asterisk are outside the scope of this application protocol.

The definitions given in this annex do not supersede the definitions given in the main body of the text.

##### E.1.1 CAD model:

The computer internal description of product shape produced by a computer aided design system.

**E.1.2 Calculation\*:**

The process of performing a mathematical computation.

**E.1.3 Calculation system\*:**

A system which produces a mathematical model of some aspect of the physical structure or behaviour of a product.

**E.1.4 Concept development\*:**

The process of producing the conceptual design.

**E.1.5 Conceptual change request\*:**

A request for a change in the conceptual design of a product.

**E.1.6 Conceptual design\*:**

The preliminary definition of the product with respect to technical, economic, ecological and strategic requirements.

**E.1.7 Definitional change request\*:**

A request for a change in the definition of a product.

**E.1.8 Design**

The development of the shape of a product taking account of its functionality and physical properties.

**E.1.9 Design change request:**

A formal request to change any form, fit or function aspect of an existing design.

**E.1.10 Development tool\*:**

A computer tool which assists in the development of a product definition.

**E.1.11 Evaluation\*:**

The testing and validation of a product design, or the prototype of a product, against its requirements.

**E.1.12 FEA system \*:**

A computer system for the analysis of CAD models with respect to such properties as stress, dynamic behaviour, heat transfer using the finite element method.

**E.1.13 Holography\*:**

A process for producing three dimensional images using lasers.

**E.1.14 Knowhow\*:**

The knowledge and experience of the people involved in the development process.

**E.1.15 Machine tool\*:**

A mechanical device, usually for cutting metal, used in the manufacturing process.

**E.1.16 Machine control code\*:**

A computer sensible code controlling a machine tool.

**E.1.17 Management\*:**

The process of controlling the use of manpower and resources.

**E.1.18 Manpower\*:**

The people, or human resources, involved in the process.

**E.1.19 Manufacturing\*:**

The realisation of the actual product.

**E.1.20 Marketing strategy\*:**

The planned method for selling the product.

**E.1.21 Material\*:**

The matter from which the product is manufactured.

**E.1.22 Measured data\*:**

Data which is obtained by measuring a product or physical model.

**E.1.23 NC-machining\*:**

A manufacturing process of material removal using a numerically controlled machine tool.

**E.1.24 Part design:**

The process of defining the form, fit and function of a part.

**E.1.25 Physical model\*:**

A representation of the product or part made of some material substance.

**E.1.26 Planning tools\*:**

Methods and equipment for the support of planning tasks, e.g. process planning systems.

**E.1.27 Prototyping\*:**

The process of producing a first example of a product for evaluation.

**E.1.28 Product plan\*:**

All available information about the manufacturing process for a product, including the sequence of manufacturing operations.

**E.1.29 Product requirement specification\*:**

The collection of all known requirements and constraints which specify the functionality and characteristics of a product.

**E.1.30 Product specification:**

The description of the characteristics of a product.

**E.1.31 Production planning\*:**

The process of producing the product plan.

**E.1.32 Quality assurance\*:**

The process of ensuring that the product meets its specification.

**E.1.33 Shape design:**

The process of developing the precise specification of the geometric form of a product.

**E.1.34 Sketch\*:**

An approximate, possibly hand-drawn, presentation of the form of a product.

**E.1.35 Simulation\*:**

The imitation of a process using a physical model or using a computer model.

**E.1.36 Standards\*:**

Formal documents, which may be international, national or company specific, which influence the design process to ensure commonality between products.

**E.1.37 Stereolithography\*:**

A method of producing three dimensional physical models of a product.

**E.1.38 Test plan\*:**

A plan for testing a product design, or prototype, to ensure that all requirements are met.

**E.1.39 Visualisation:**

The pictorial representation of a product.

**E.1.40 Visualisation system\*:**

A system which processes the product description data into a pictorial form of representation.

## **E.2 Mechanical design requirements for model exchange**

The following provides an introduction to requirements for data exchange in mechanical design and engineering in general. Background information is given on where surface model exchange specifications are expected to be utilized within that field.

### **E.2.1 Information related to mechanical design and engineering**

The mechanical design area has a large scope that is limited here to the information required for the representation of surface models. The following gives an overview of types of information mechanical design deals with.

#### **E.2.1.1 Shape description (geometry and topology)**

Depending on their applications shape representations of mechanical design part models may be of different levels of completeness and enforced by appropriate constraints. Therefore, different model design methods are applied.

- volume based design (B-rep, CSG),
- surface based design,
- wireframe based design.

The resulting representations differ from each other by model quality criteria such as completeness, conciseness, degree of freedom, complexity. A product may be described using one or several of these representations. In general product characteristics determine which of the representations would be the most appropriate one. Conversions from one to another representation happen - in both directions. This part of ISO 10303 applies to alternative two in the list above, surface based design. Both volume representations and wireframe representations may be converted into appropriate conformance classes of this part.

#### **E.2.1.2 Physical property description**

A variety of properties may be attached to a part model. Some of them, e.g. material, are assigned during the design process. Other physical properties such as surface roughness and surface treatment may first be determined at a later stage in the life cycle of the part.

#### **E.2.1.3 Part description**

The descriptions of parts contain aggregations of administrative and structural data. These data link together shape, material and other aspects of the part. Also this information may be dealt with during the design process.

### **E.2.1.4 Assemblies**

Assemblies and sub-assemblies are results of product design. Corresponding data needs to be forwarded to both engineering and manufacturing.

### **E.2.1.5 Product information description**

Product information description covers basic administrative data for managing a product through the design process and further to engineering and manufacturing. Information comprised is e.g.:

- product identification (name),
- ownership (company, department, designer, manager),
- history (date of design, updates, release status, revisions).

## **E.2.2 Surface model exchange in mechanical design**

This sub-clause describes where and when, in the processes of mechanical design and engineering, information on surface models is exchanged.

The outline of a product is primarily fixed in the styling and conceptual design phases. Non-computerized tools dominate. The final product shape is typically detailed by the Department for Detail Design in utilising CAD-systems.

The design data model - represented as a CAD data model - is transferred to a heterogeneous CAD environment in a system-to-system approach by means of neutral files. The model may be transferred to other departments within the same company or to other companies, for further design, for analysis, for manufacturing, or other purposes. The scenario described in the following is conceivable.

### **E.2.2.1 From design to product analysis**

In order to evaluate a design the design model is transferred to an engineering department which may perform - dependent on the functionality of the product - e.g. stress or thermal analysis. If the design does not satisfy the requirements, it may be modified. A modification proposal for the design data will be sent back to the design office (or kept in a database). The designer will get a notice on the requested change.

### **E.2.2.2 From design to design (different companies)**

Very often the manufacturing of complex products is done with parts from different design and manufacturing sources. In this case the designs of individual parts must be exchanged across company borders. This is common in surface design, where parts from many different sub-contractors must fit e.g. into a given car body shape.

### **E.2.2.3 From design to manufacturing planning and manufacturing**

A detailed manufacturing process plan is required in order to manufacture a product. For metal cutting such a plan would select appropriate technologies such as milling, drilling, turning and grinding. Numerically controlled (NC) machine tool programs are generated. For this NC-generation process, data is derived from corresponding CAD-models (required data may concern geometry, topology, material, product structure). The NC tool paths are produced by means of NC-programming systems. Collision checks against machine tools are performed and tool paths simulated. The results are NC-machine programs that can be down-loaded to appropriate types of NC-machines. A similar procedure is adopted for the programming of robots (e.g. for feeding of NC-machines). There, however, surface models are seldom utilised.

### **E.2.2.4 From design to assembly simulation**

If robots are applied to assemble products, robot simulation programs may be employed in order to visualise the assembly process prior to its physical execution. Assembly path simulations and collision checks are performed based on geometric assembly paths which are typically generated interactively at a simulation workstation. For this purpose product shapes are used from corresponding CAD-models. Assembly simulation mainly requires Brep-models, as automatic collision controls can only be done by solid modellers. Sometimes, however, visual control is sufficient. In such cases surface models are also involved.

### **E.2.2.5 From design to quality assurance**

Quality assurance controls design, engineering and manufacturing processes. Design data is often referenced by quality assurance activities subsequent to the design process.

EXAMPLE 10 – The design model is e.g. a reference for manufacturing inspection. After a mechanical part has been machined, its real shape is evaluated by a coordinate measurement machine. If there is a deviation beyond a certain tolerance limit between the measured points and the ideal CAD-model points, the manufactured part will not pass.

### **E.2.2.6 From design to documentation**

The design master model, resident in a computer and based on topology, geometry and other associated information is used to derive CAD drawings. The 2D CAD documentation typically consists of orthogonal projections of the 3D model into 2D space. The resulting CAD drawings describe the geometry of designed parts, explanatory text, dimensioning, hatching of sectioned areas, and administrative information.

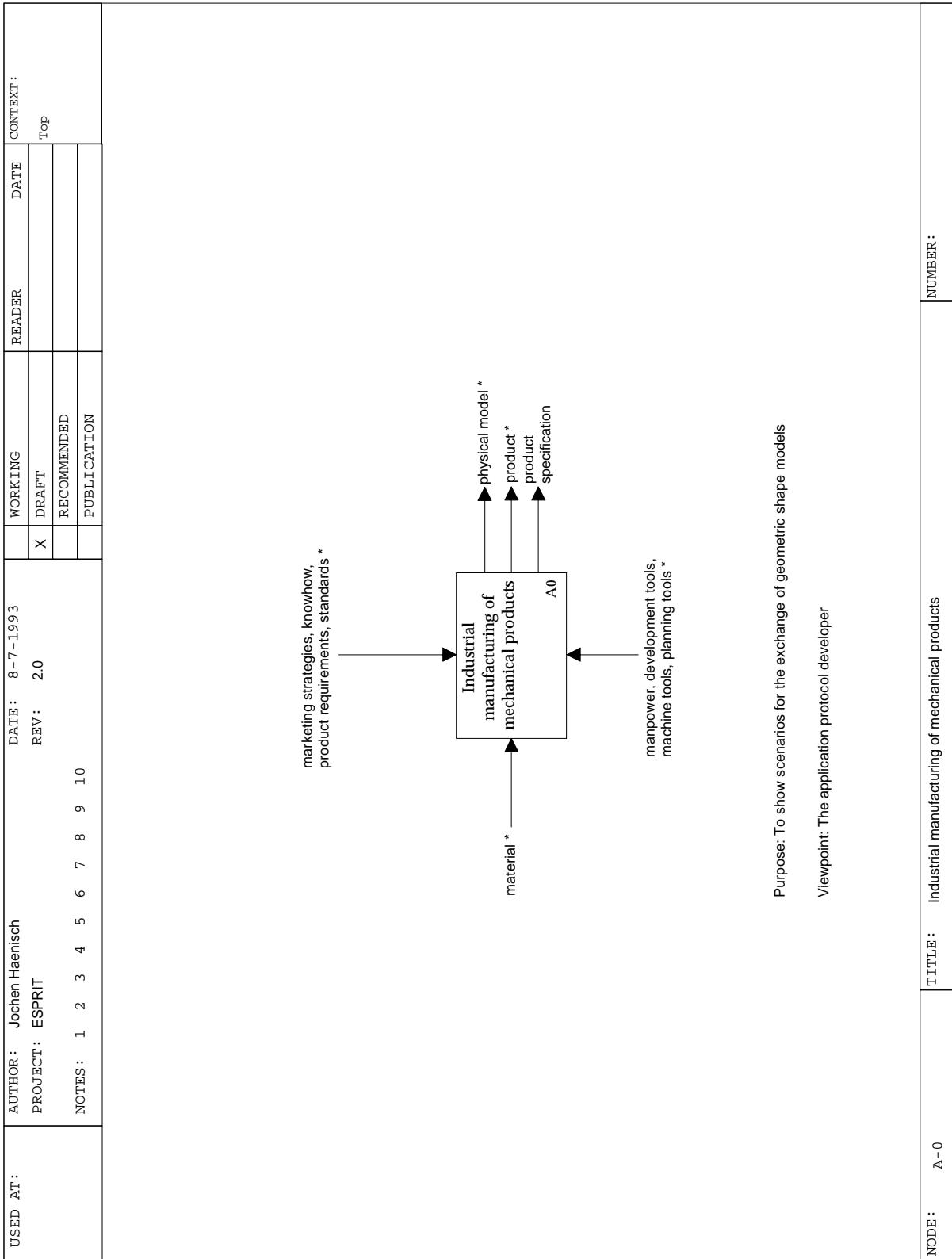
### **E.2.2.7 Feedback to the design department**

The first design of a part is in general not the final one. Model data is often transferred from manufacturing process planning teams or part analysts back to the design department where it is modified. The senders give feedback to the design office in form of change requests accompanied

by the original CAD model. Administrative information may also be involved in the design process and in its feedback loop.

### **E.3 AAM diagrams**

The application activity model is given in figures E.1 through E.4. The graphical form of the application activity model is presented in the IDEF0 activity modeling format. Activities and data flows that are out of scope are marked with asterisks.



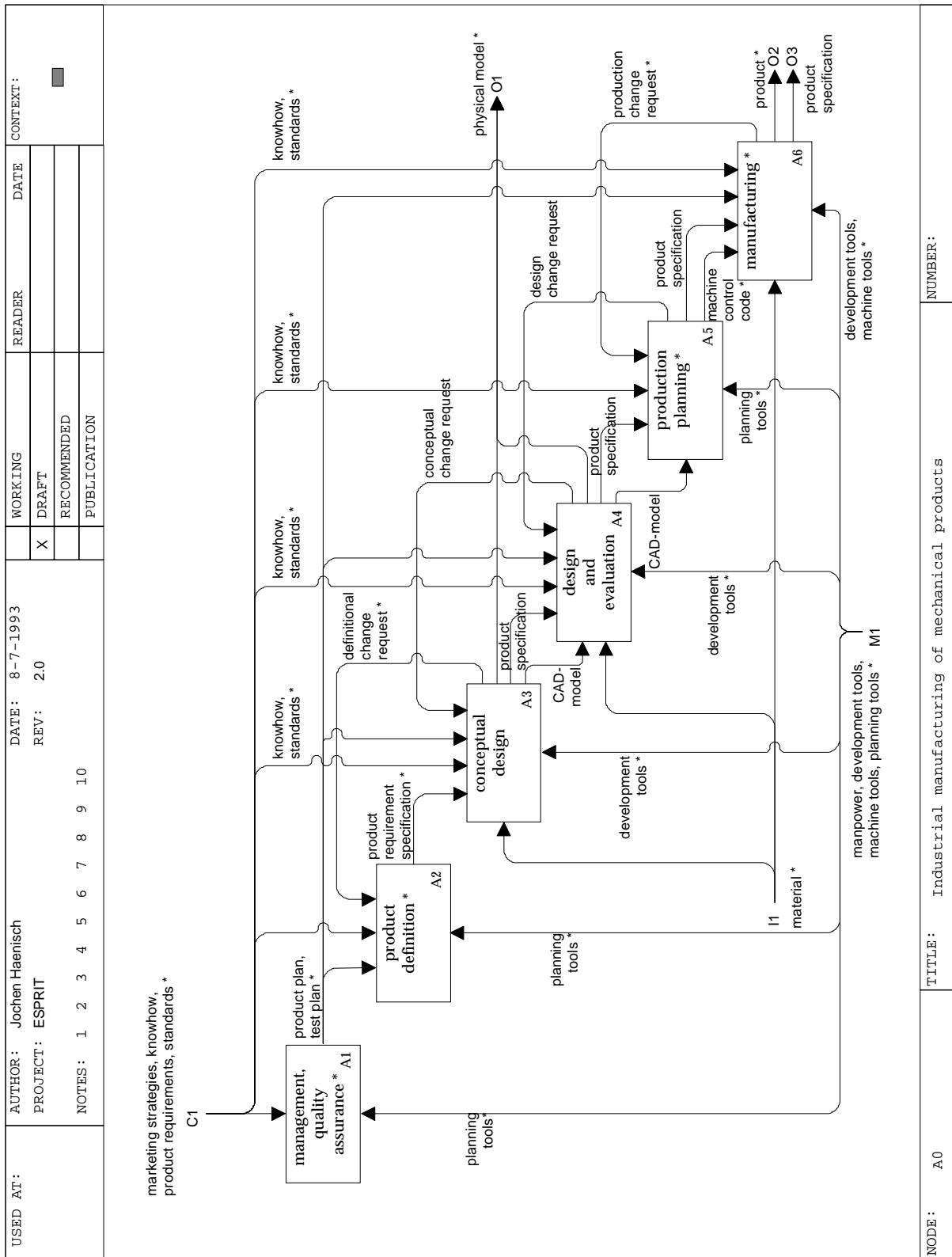
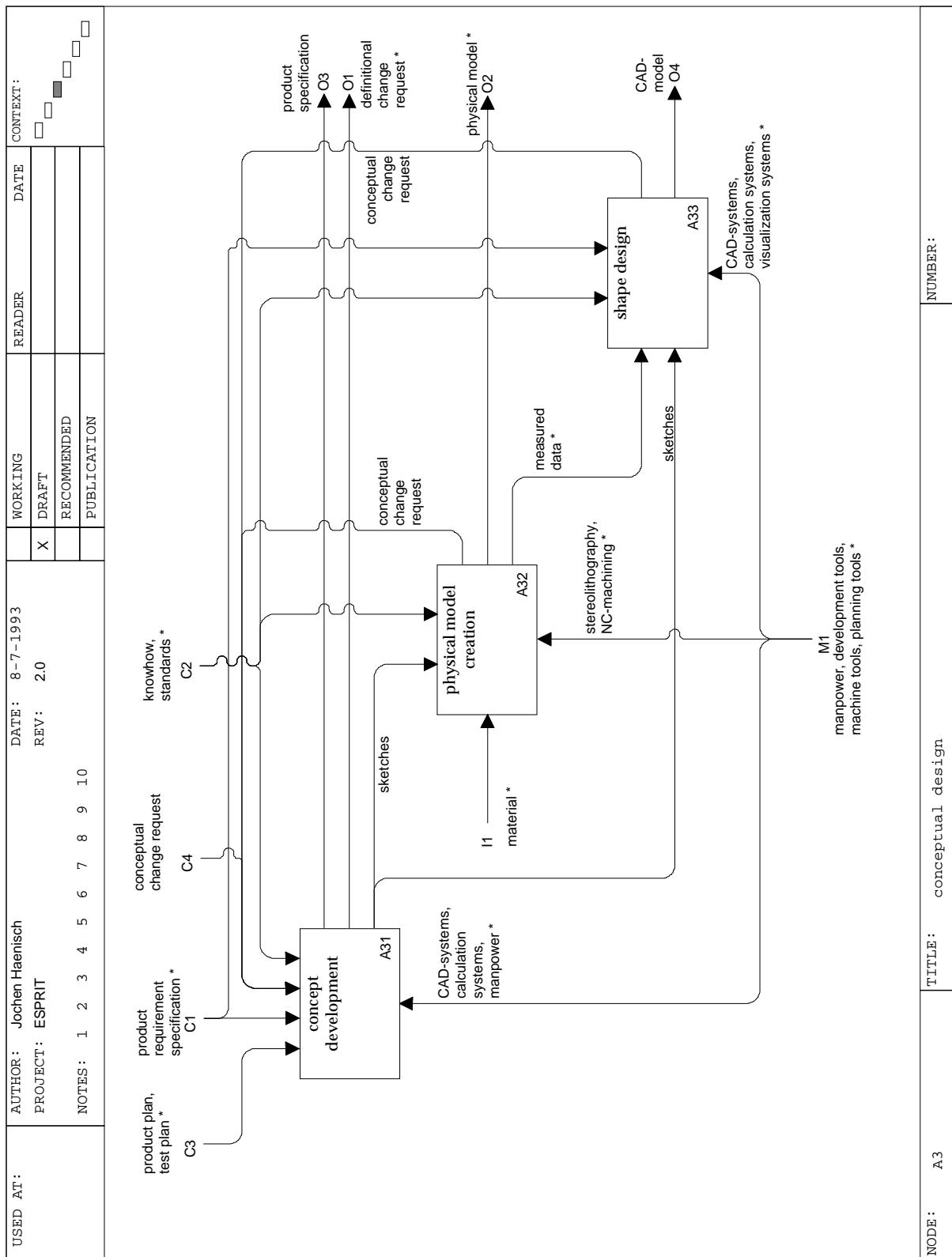


Figure E.3 Details of industrial manufacturing of mechanical products



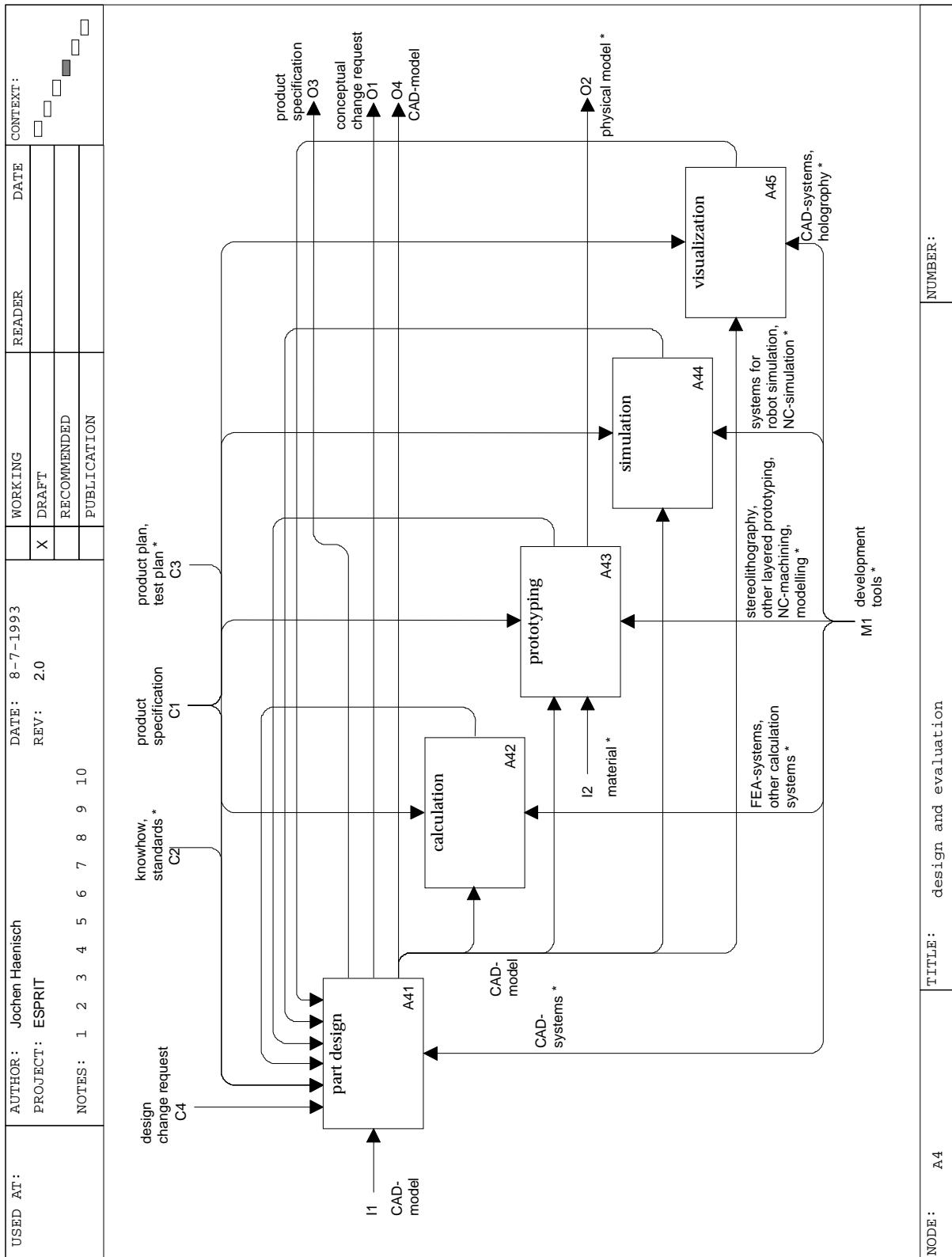


Figure E.4 Design and evaluation

**Annex F**  
(informative)

**Application reference model**

This annex provides the application reference model for this part of ISO 10303 and is given in figures F.1 through F.30. The application reference model is a graphical representation of the structure and constraints of the application objects specified in clause 4. The graphical form of the application reference model is presented in the NIAM format. The application reference model is independent from any implementation method.

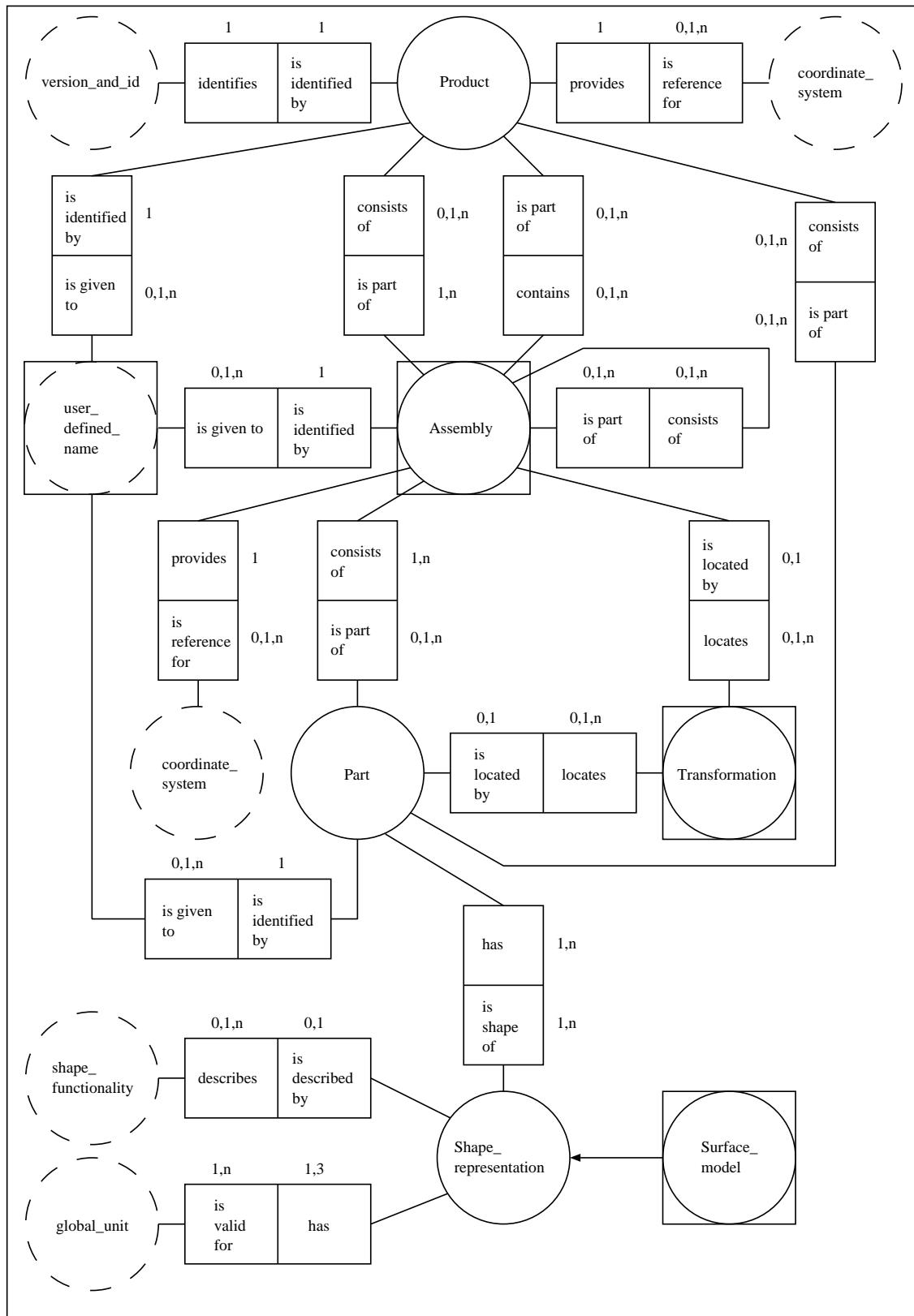


Figure F.1 – Requirements for product structure

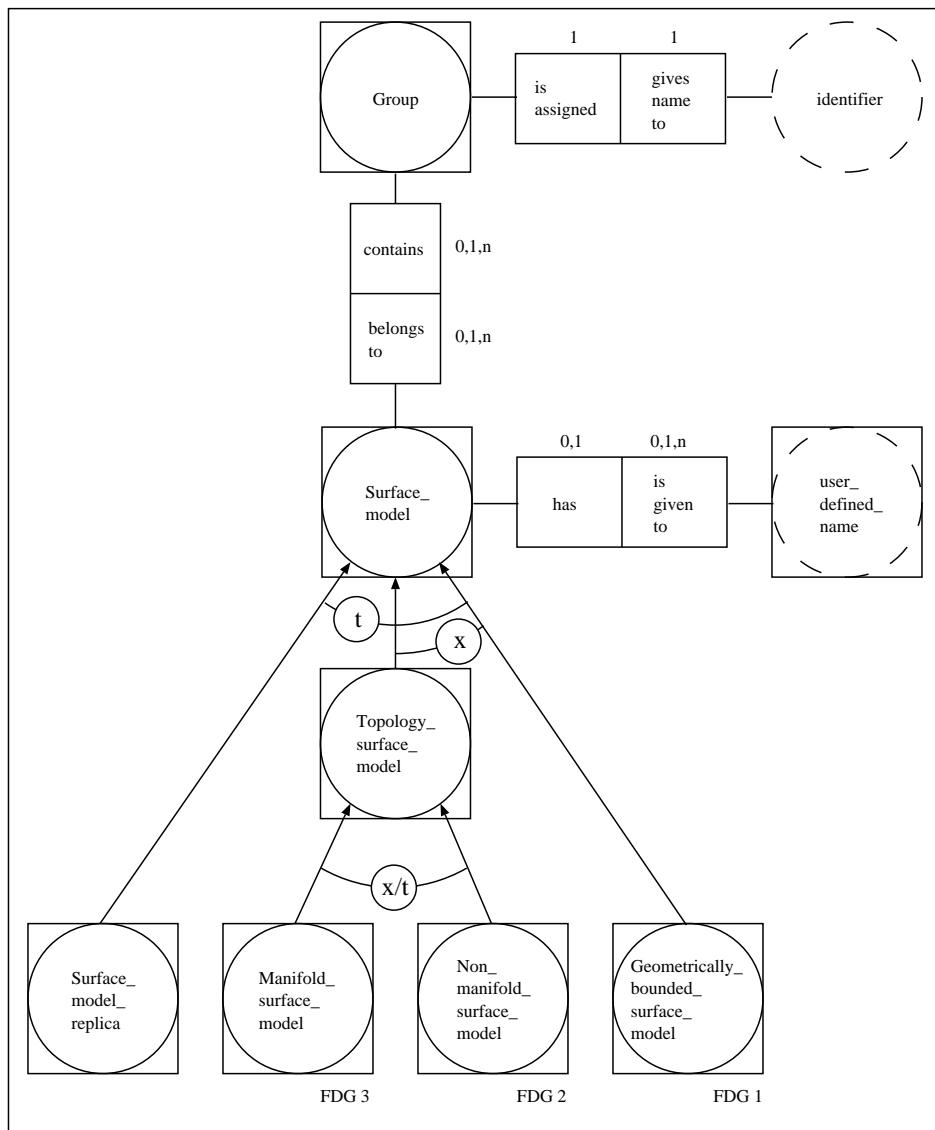


Figure F.2 – Surface model

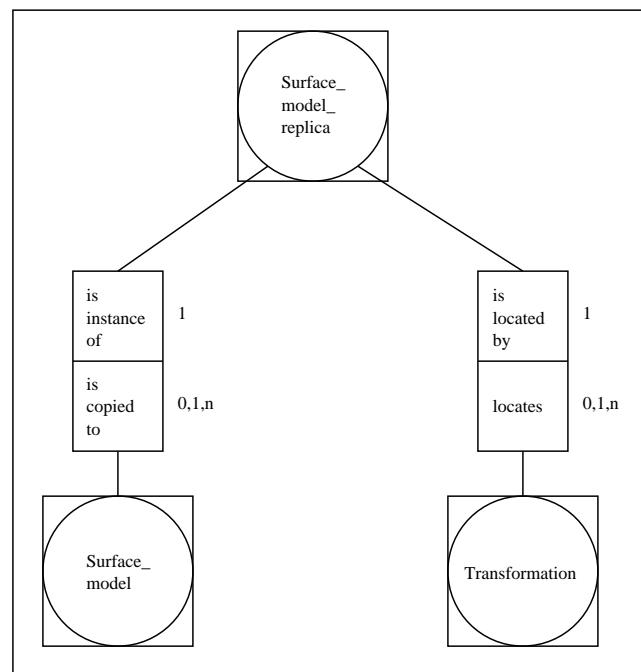


Figure F.3 – Surface model replica

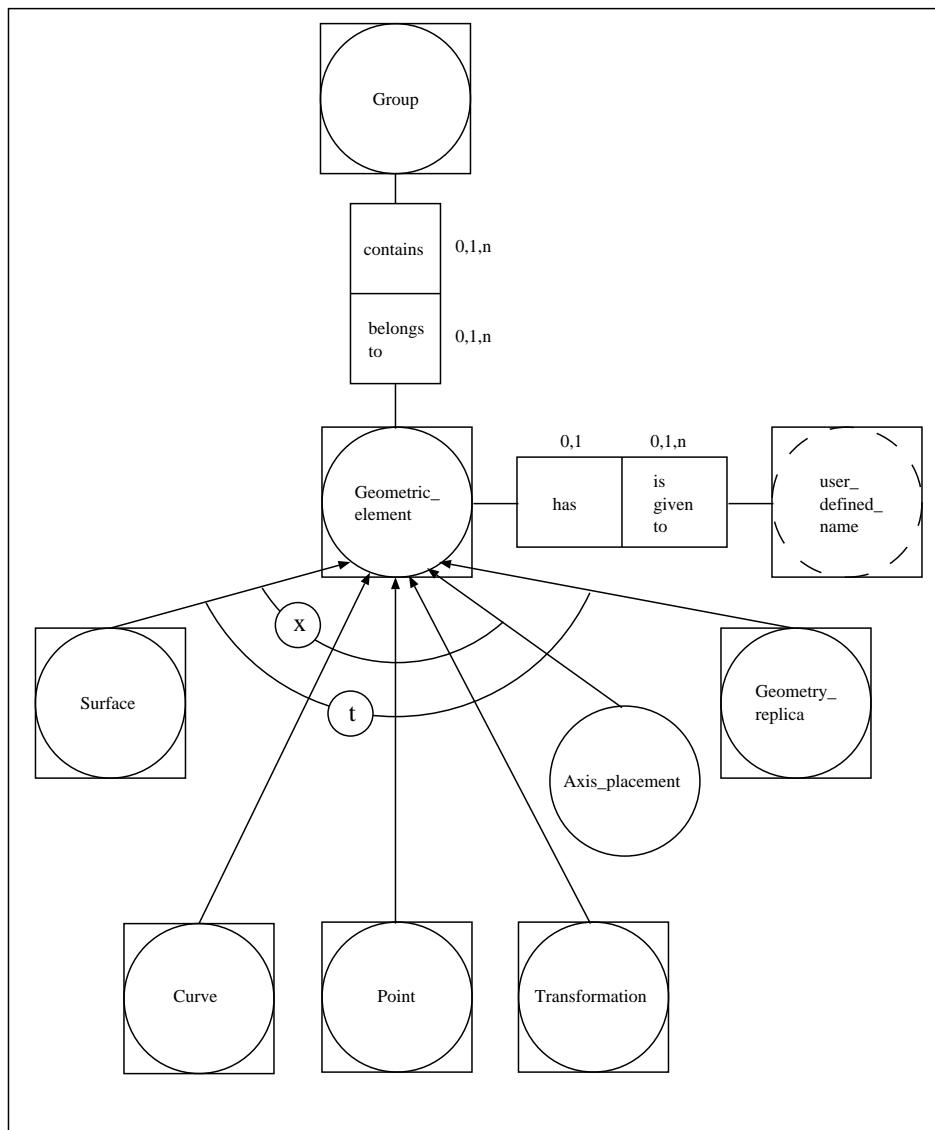


Figure F.4 – Geometry

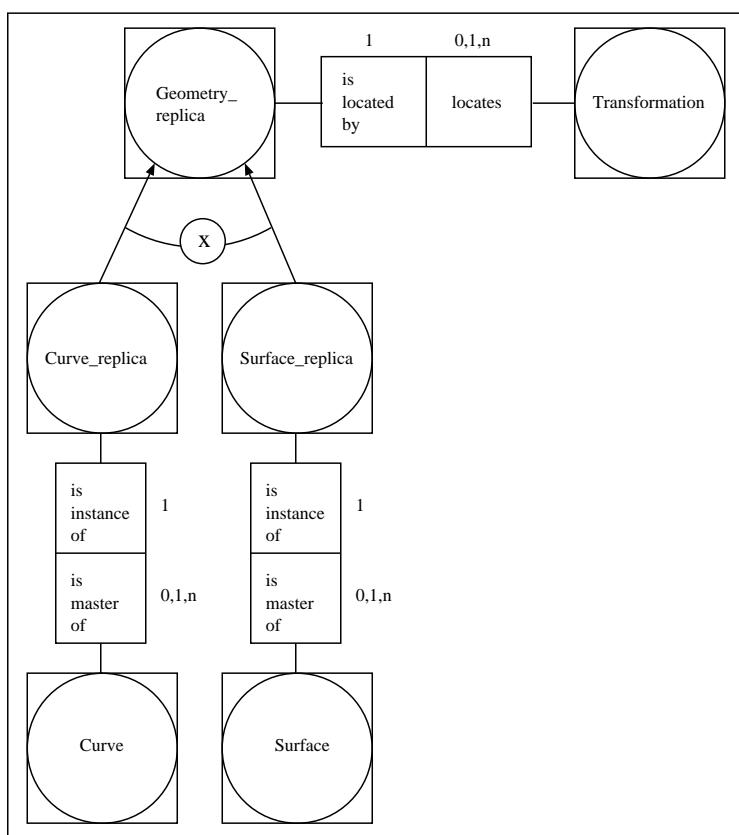


Figure F.5 – Geometry replica

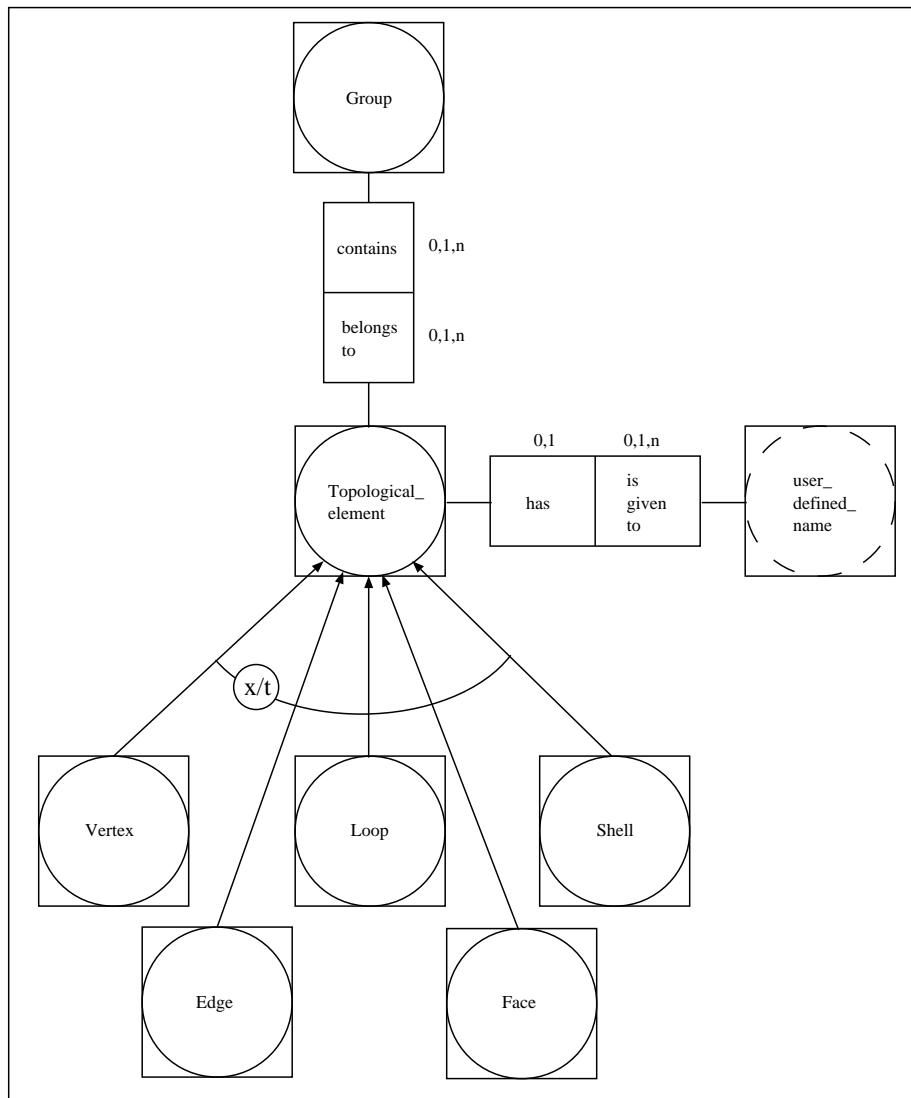


Figure F.6 – Topology (only FDG2 and FDG3)

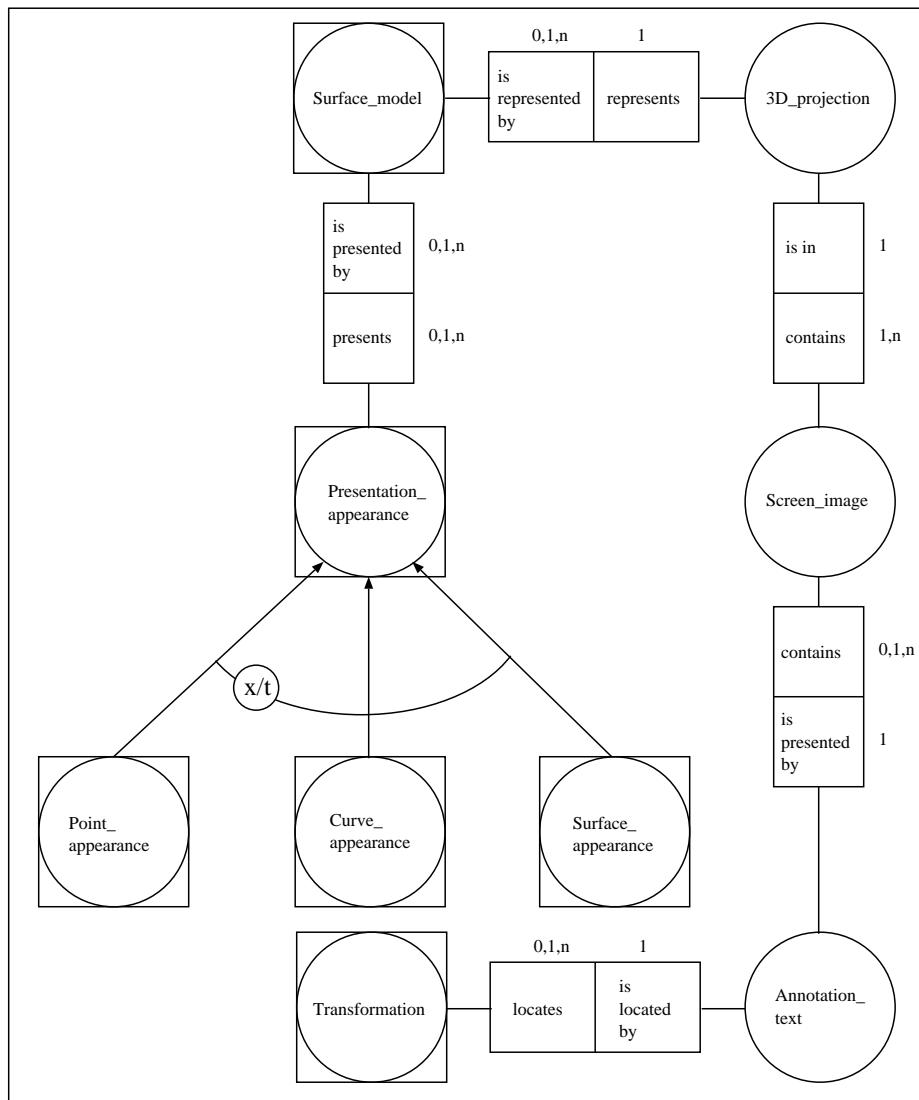


Figure F.7 – Visual presentation

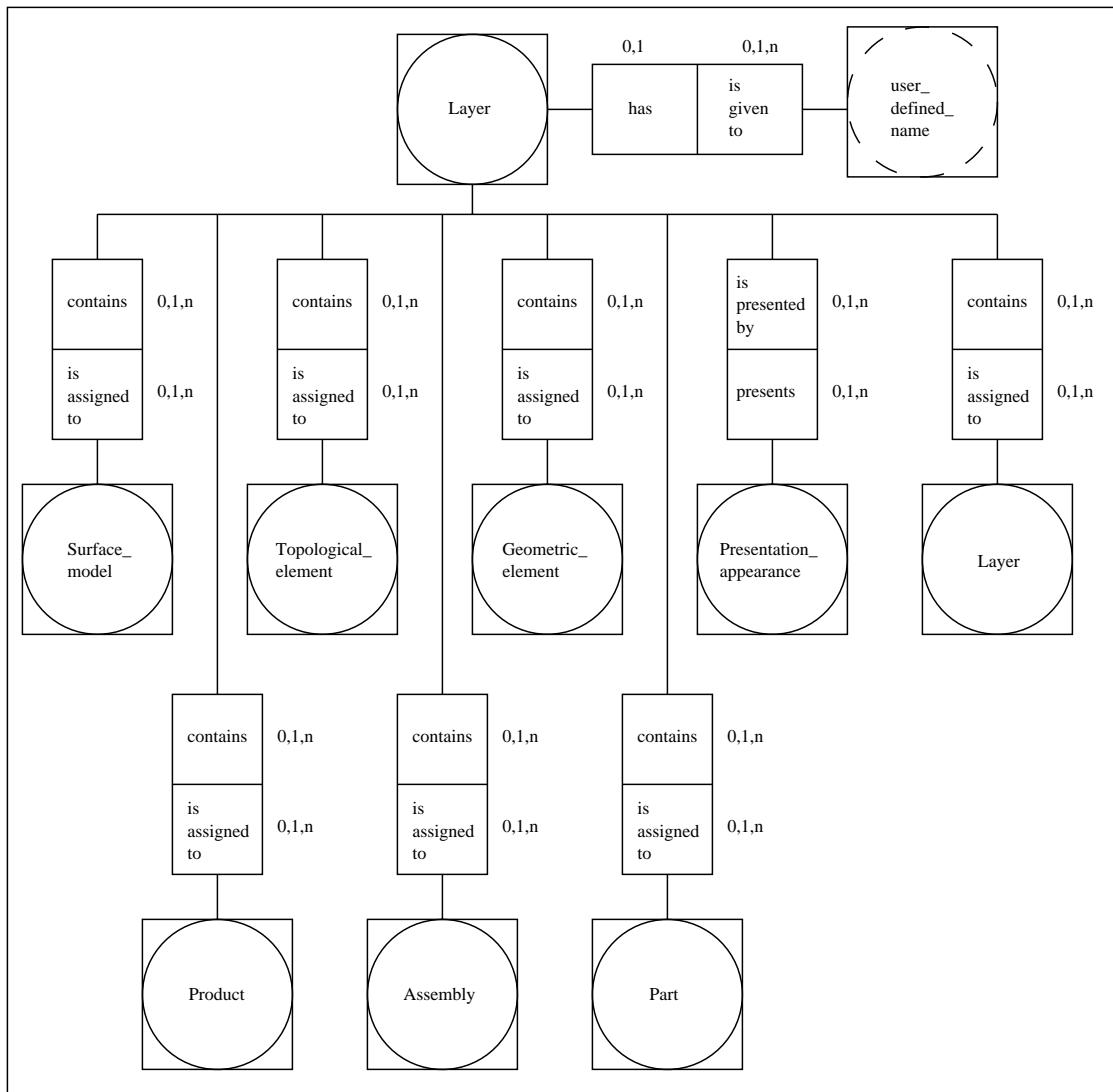


Figure F.8 – Layers

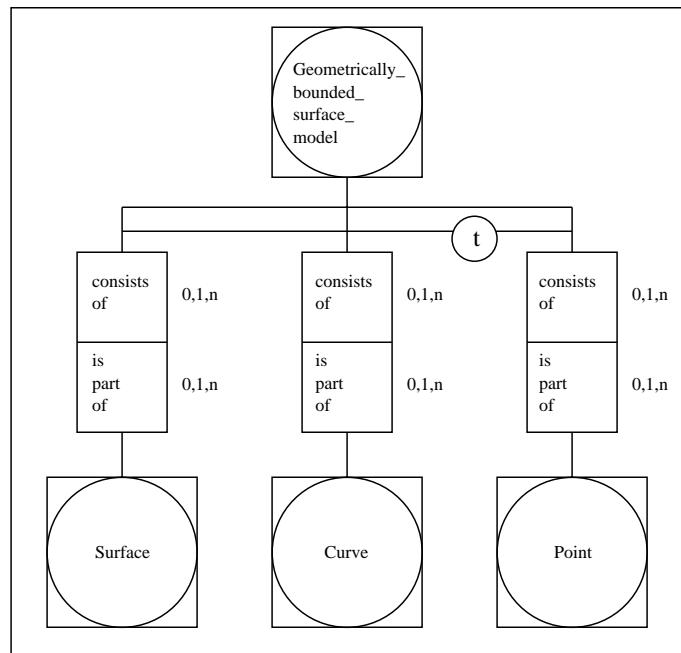


Figure F.9 – FDG1 - geometrically bounded surface model

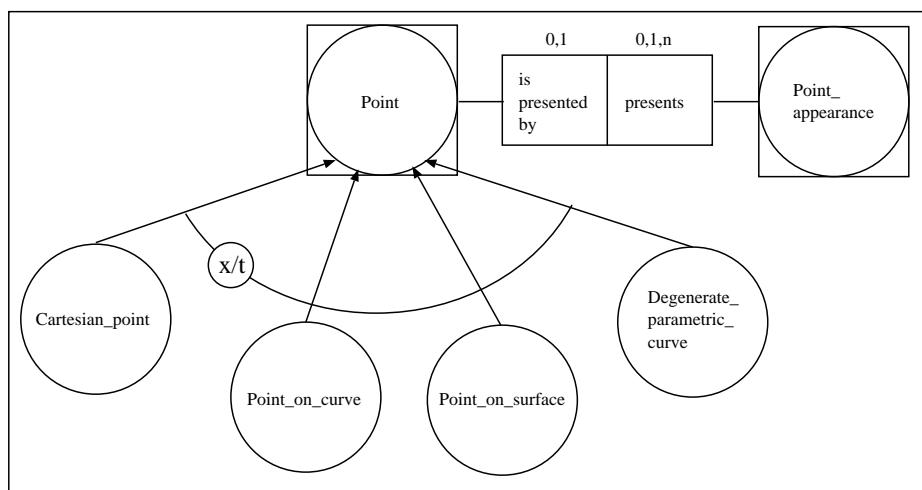
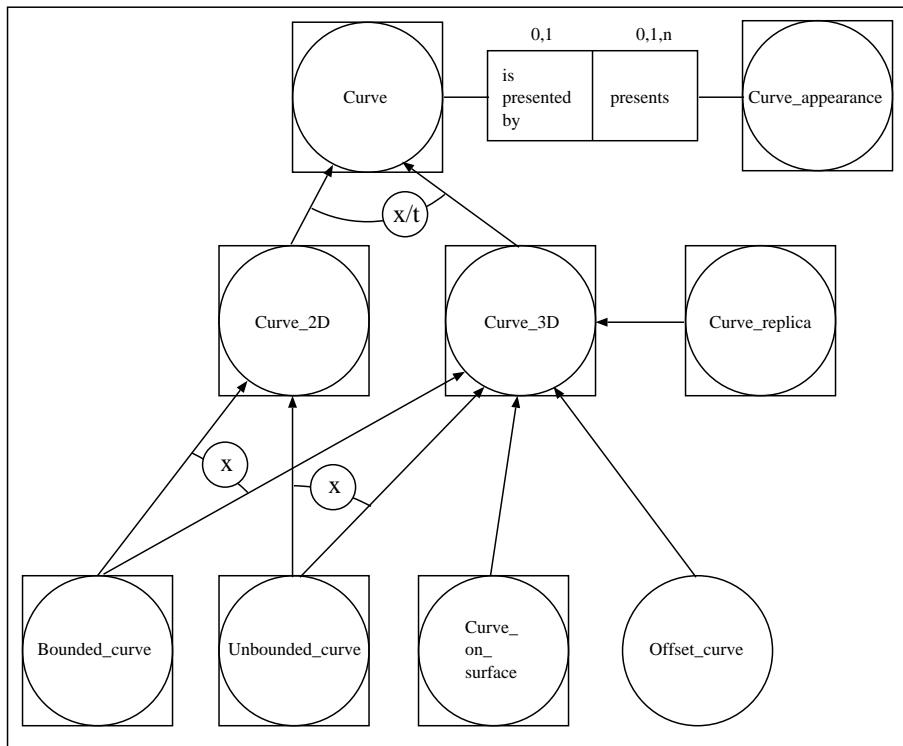
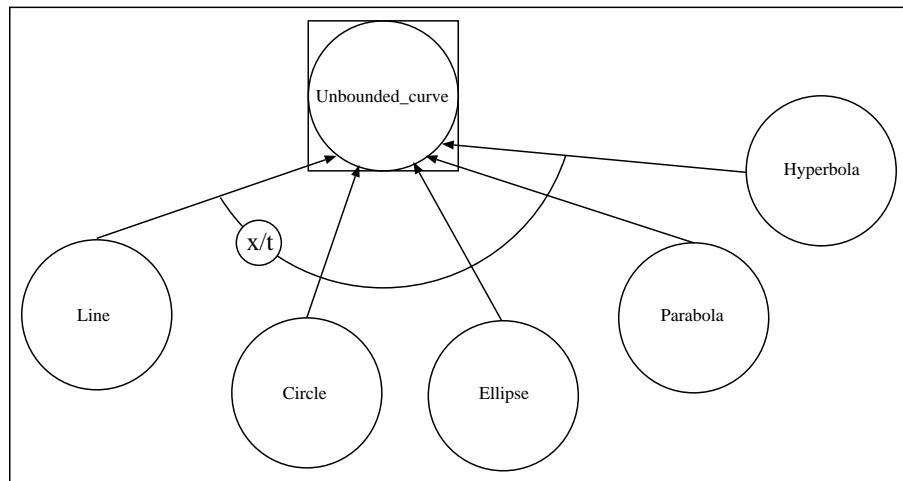


Figure F.10 – Point



**Figure F.11 – Curve**



**Figure F.12 – Unbounded curve**

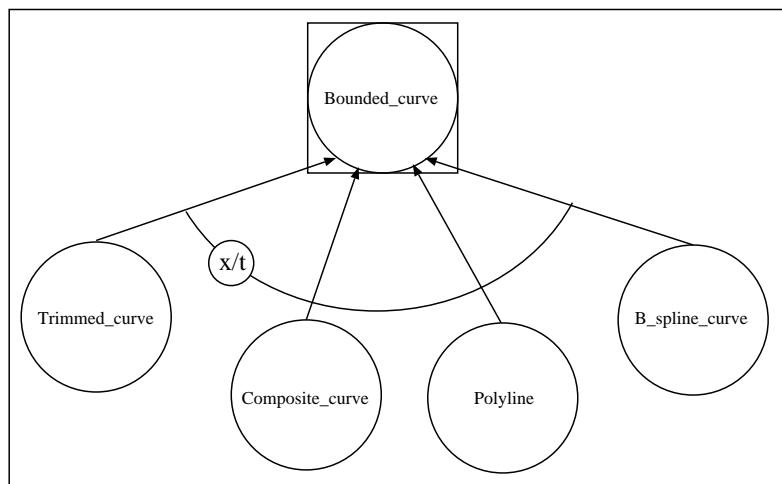


Figure F.13 – FDG1 - bounded curve

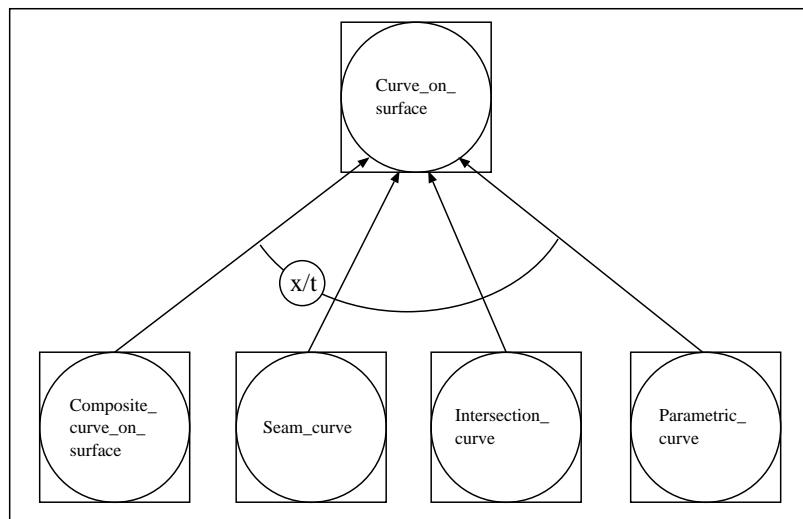


Figure F.14 – FDG1 - curve on surface

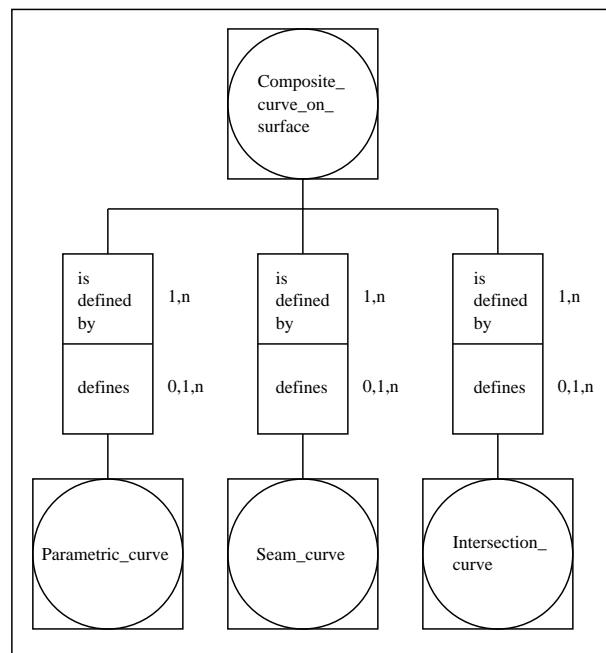


Figure F.15 – FDG1 - Composite curve on surface

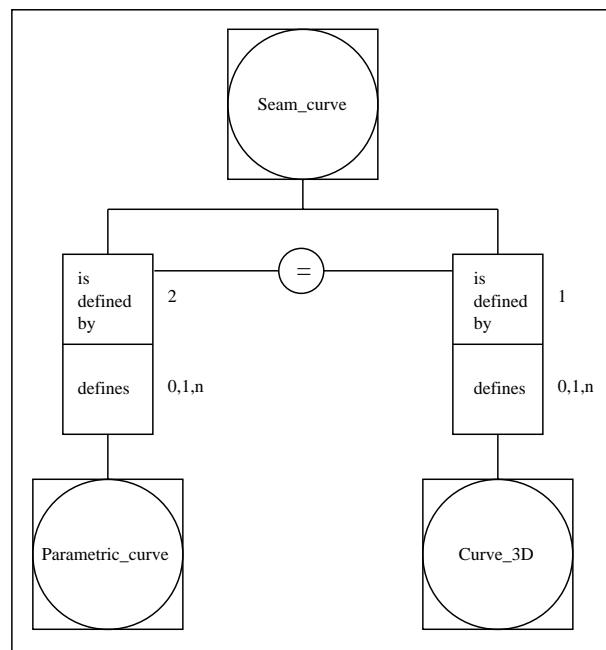


Figure F.16 – Seam curve

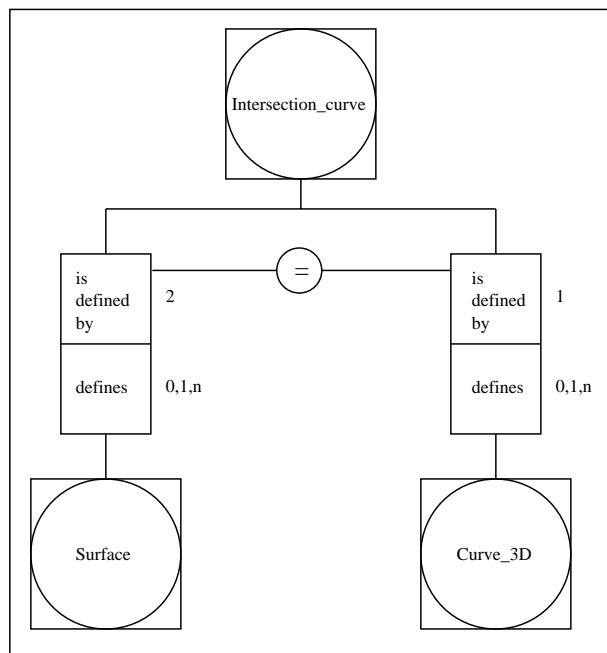


Figure F.17 – Intersection curve

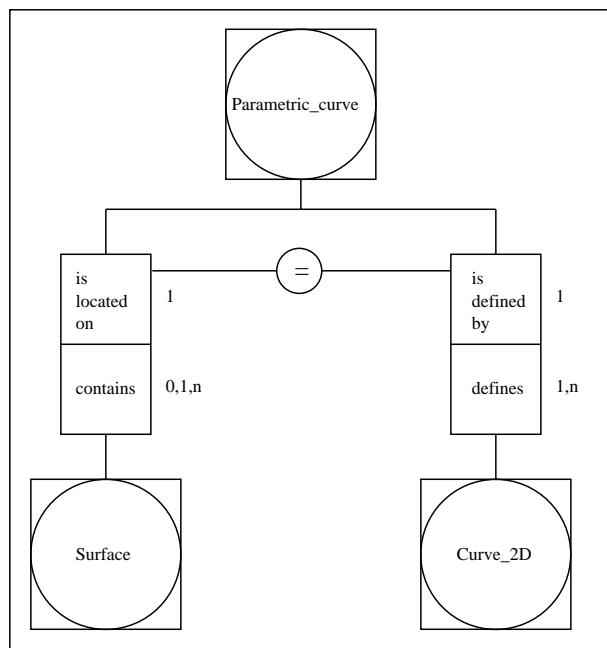


Figure F.18 – Parametric curve

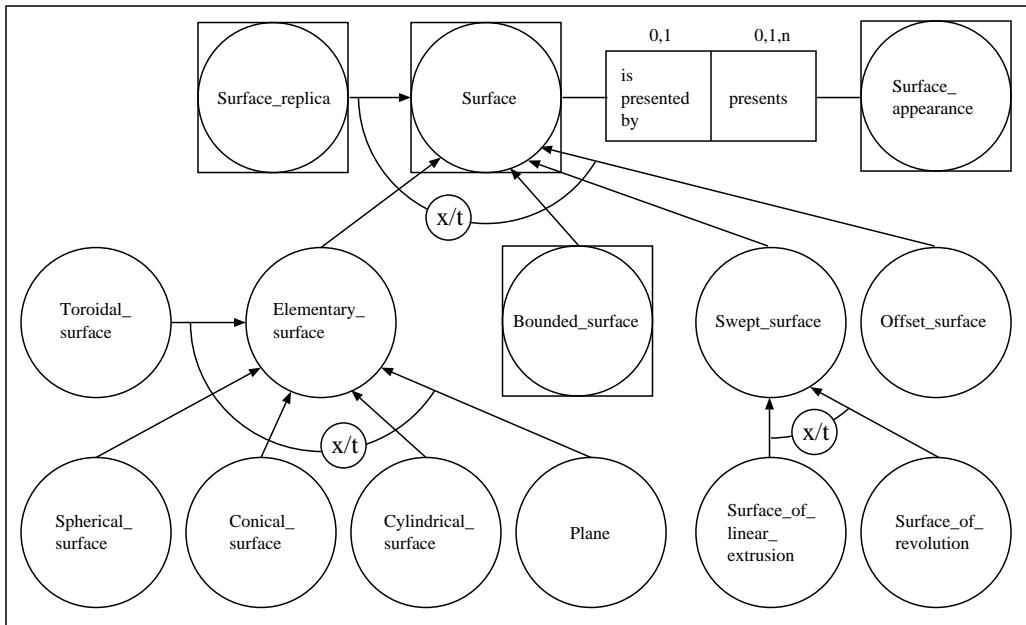


Figure F.19 – Surface

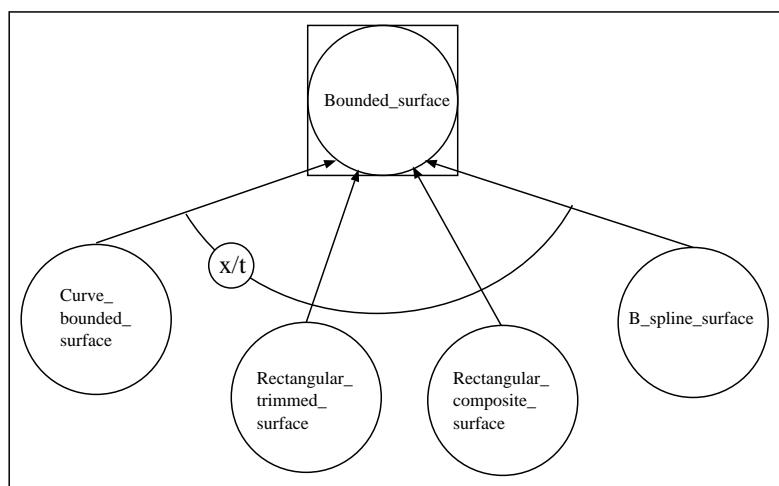


Figure F.20 – FDG1 - bounded surface

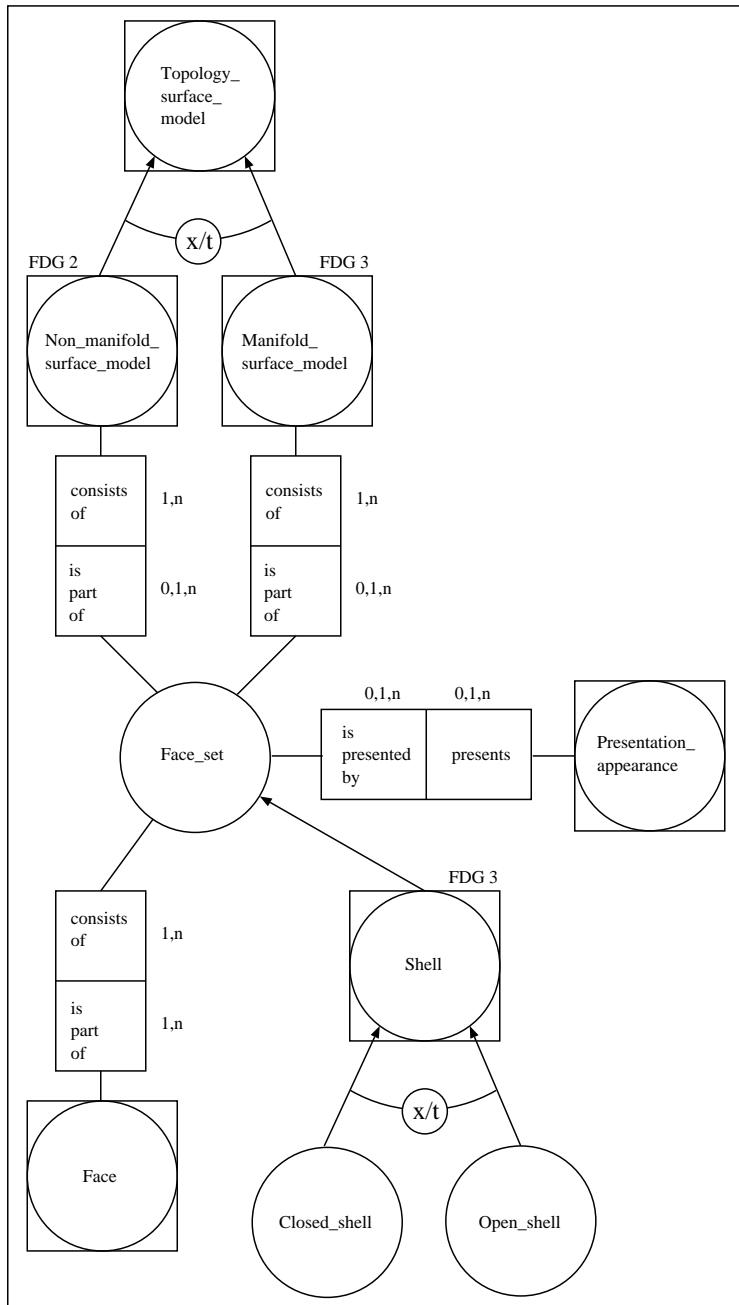


Figure F.21 – FDG2, FDG3 - topology surface models

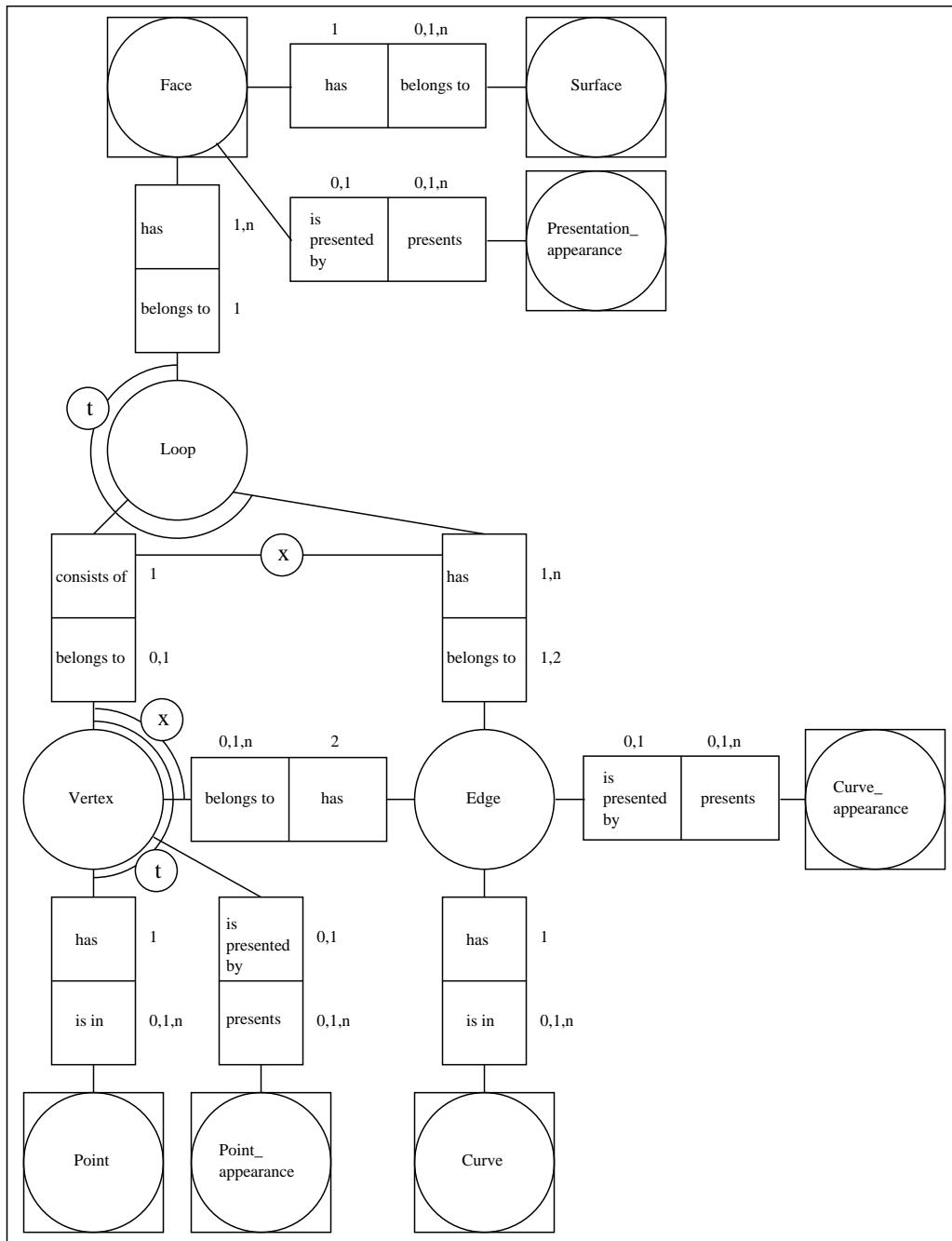


Figure F.22 – FDG2, FDG3 - face

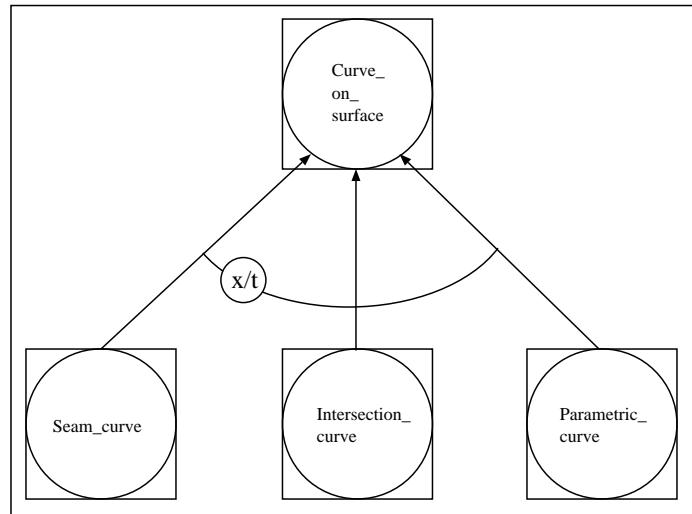


Figure F.23 – FDG2, FDG3 - curve on surface

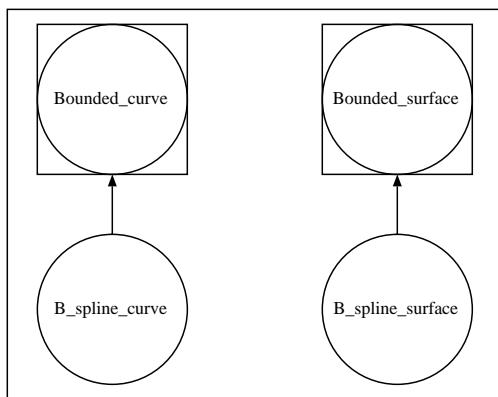


Figure F.24 – FDG2, FDG3 - bounded curve and bounded surface

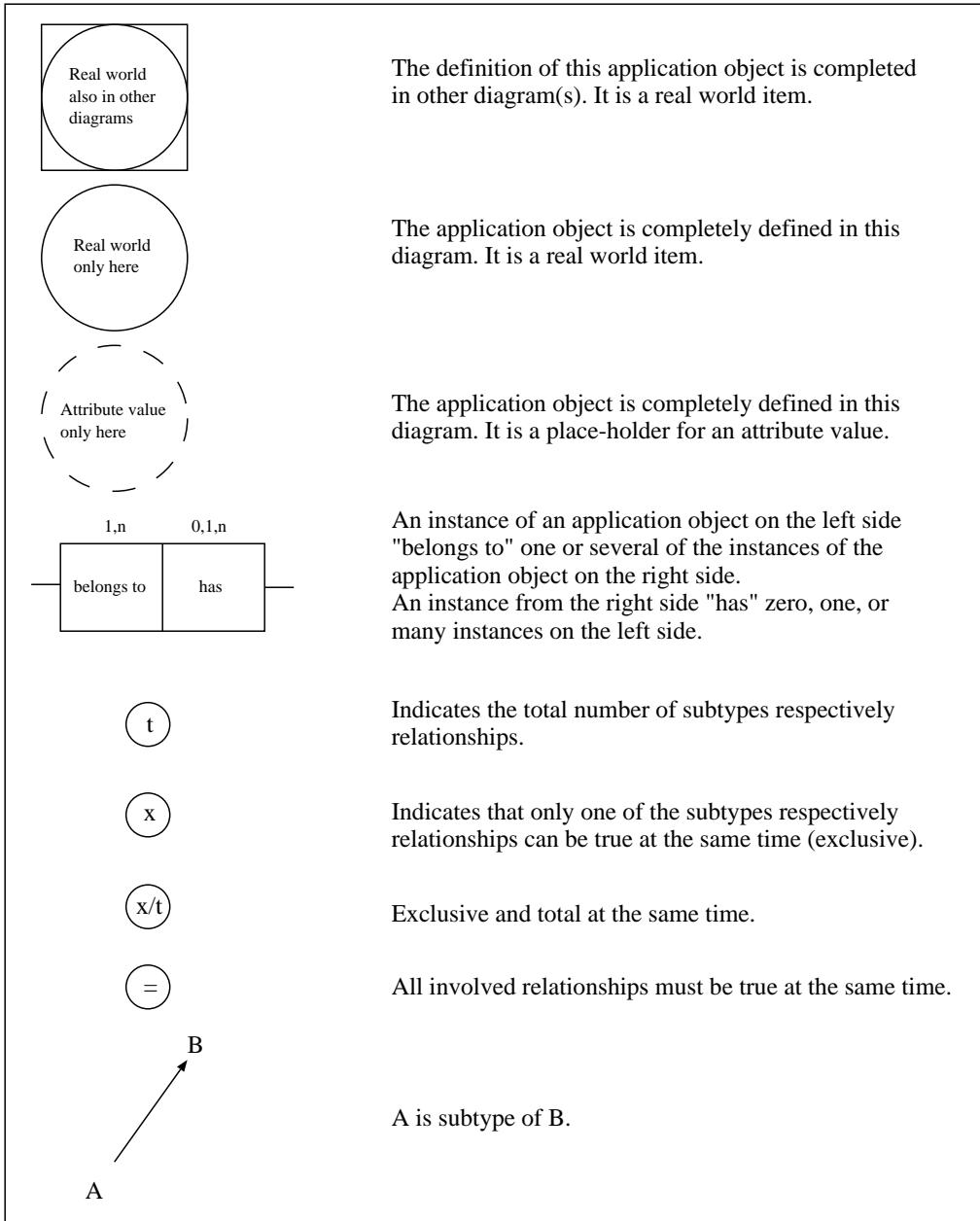


Figure F.25 – Explanations of NIAM symbols

**Annex G**  
(informative)

**AIM EXPRESS-G**

Figures G.1 through G.17 correspond to the AIM *EXPRESS* annotated listing given in annex A. The figures use the *EXPRESS-G* graphical notation for the *EXPRESS* language. *EXPRESS-G* is defined in annex D of ISO 10303-11.

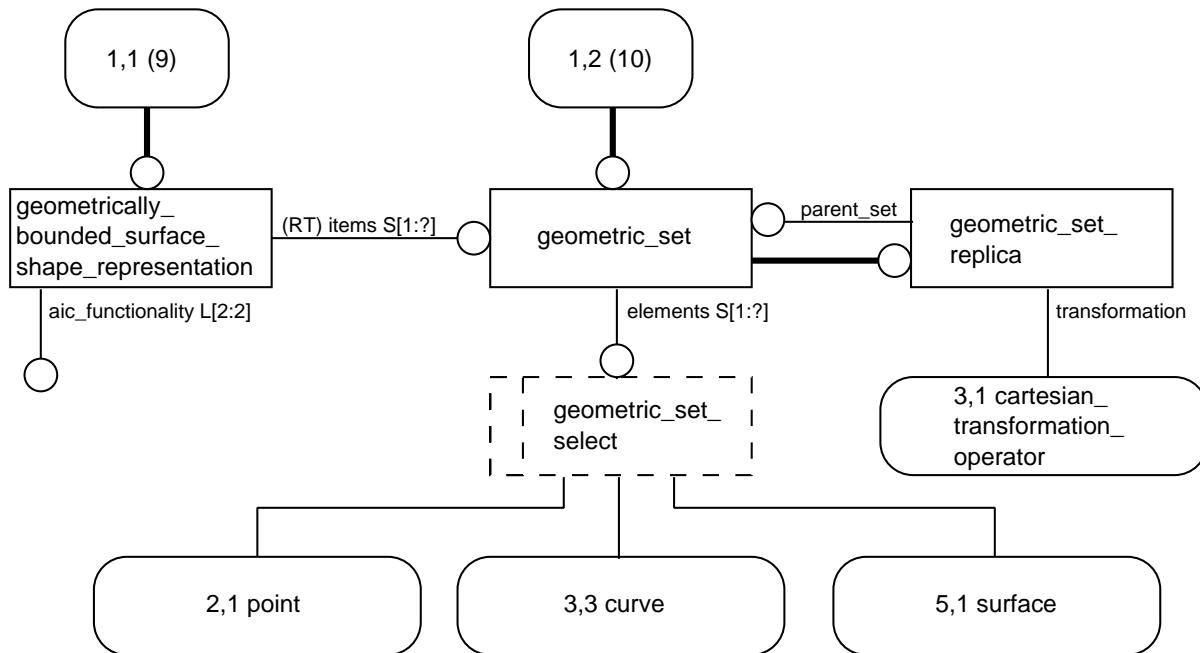


Figure G.1 – Graphical notation of the major aspects of the `geometrically_bounded_surface_shape_representation` in the `mechanical_design_surface_schema`. (See also figures G.2 to G.17)

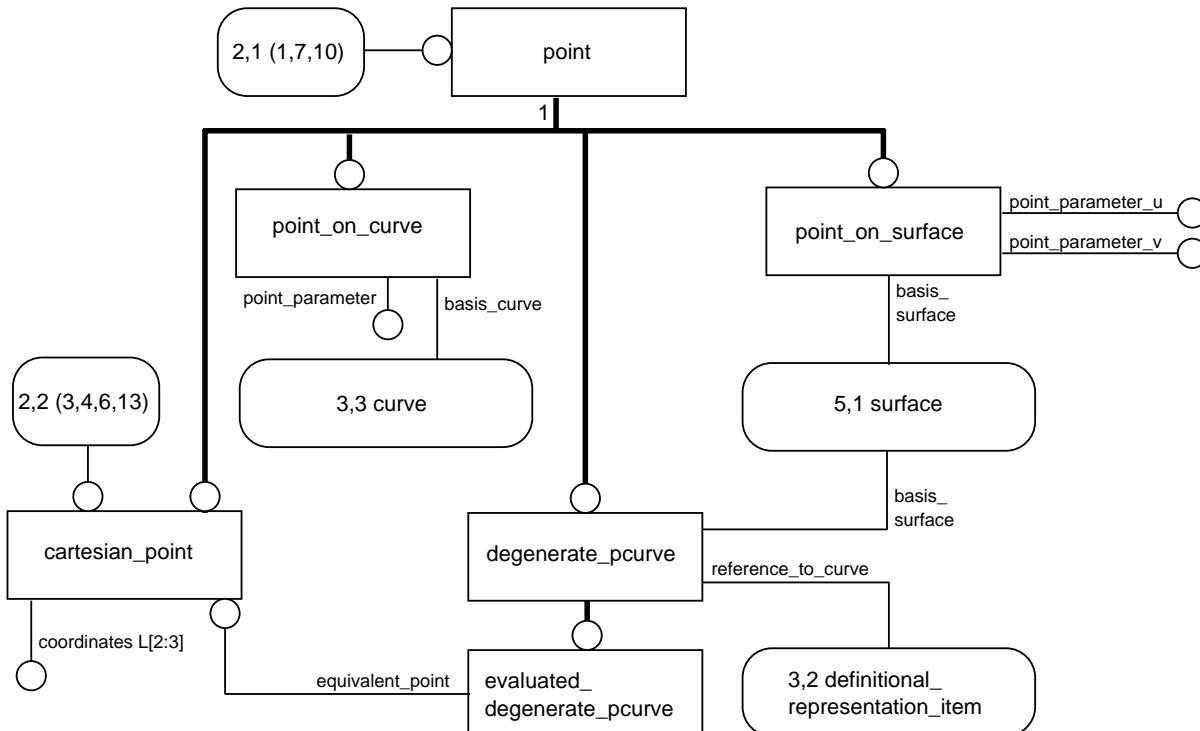


Figure G.2 – Graphical notation of the major aspects of points in the `mechanical_design_surface_schema`. (See also figures G.1 and G.3 to G.17)

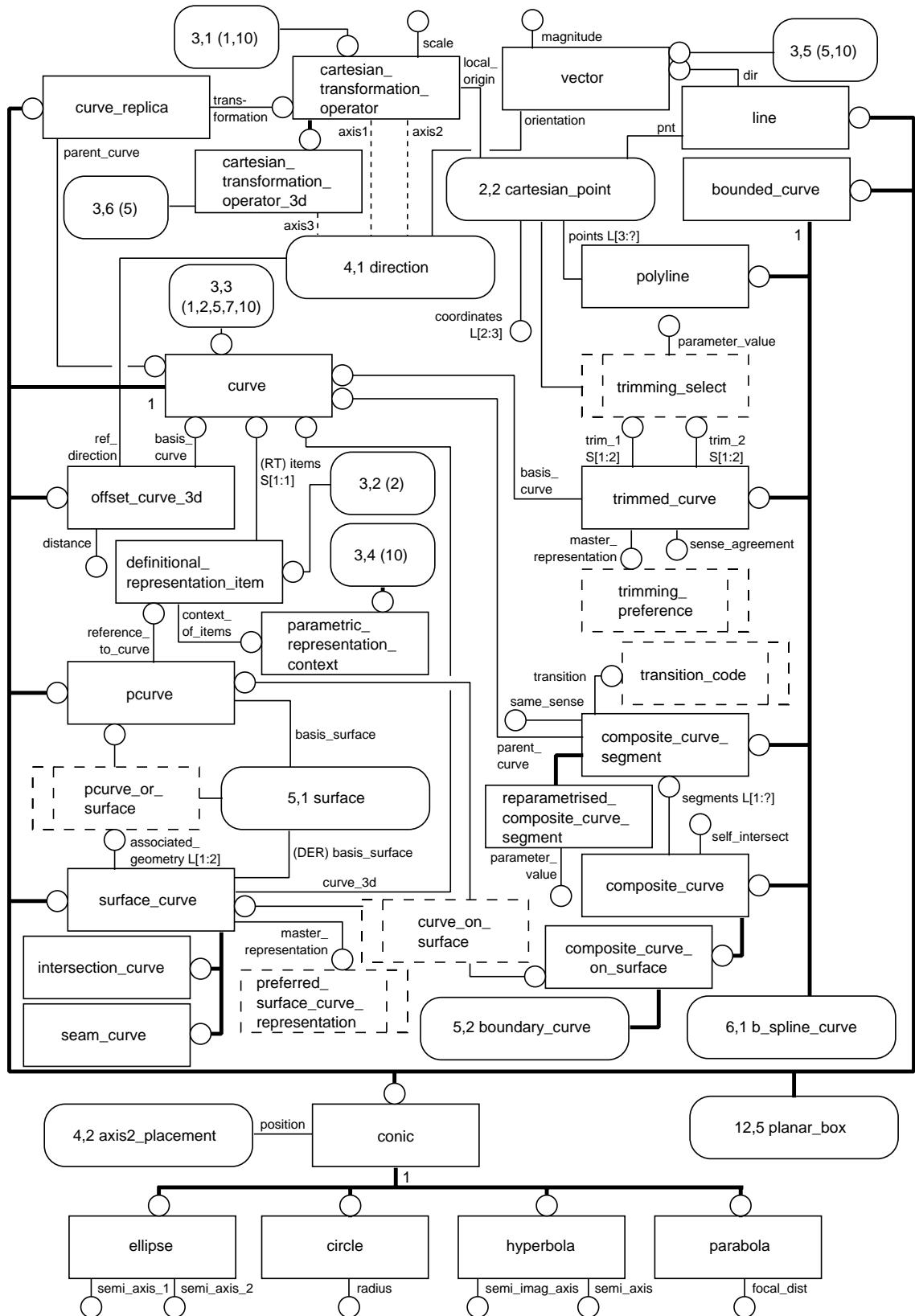


Figure G.3 – Graphical notation of the major aspects of curves in the mechanical-design-surface-schema. (See also figures G.1 to G.2 and G.4 to G.17)

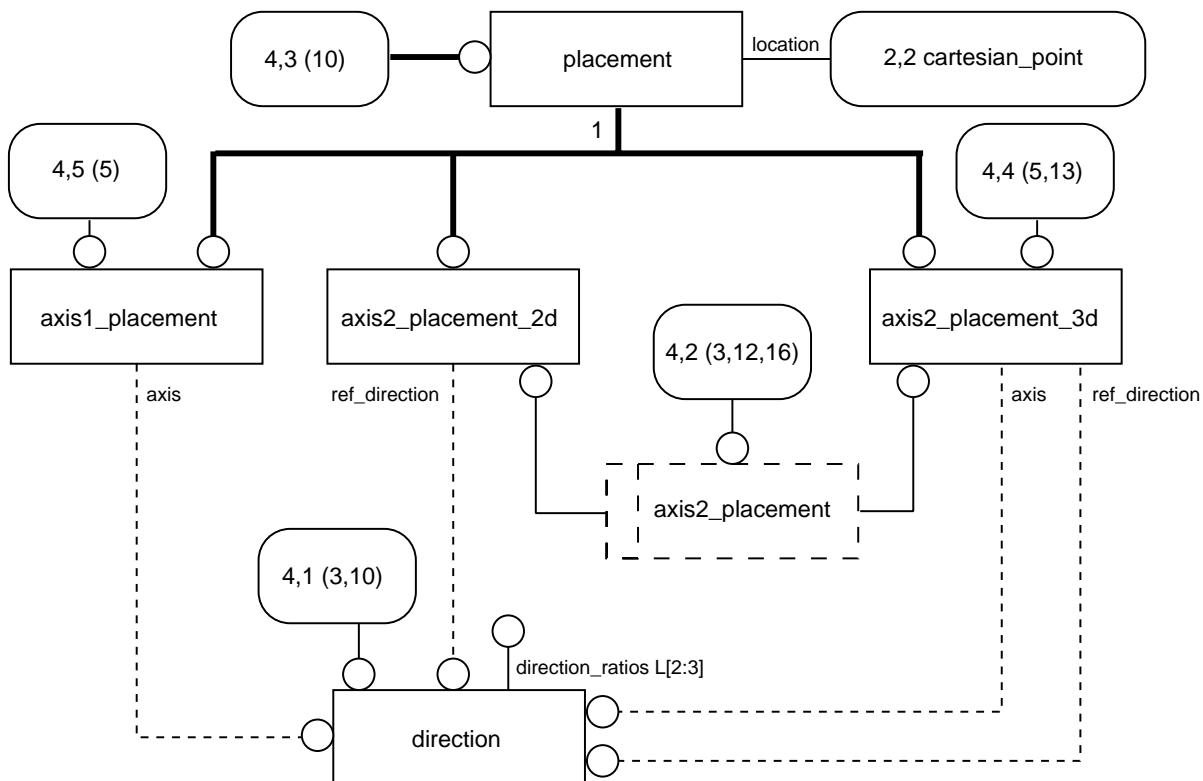


Figure G.4 – Graphical notation of the major aspects of placement in the `mechanical_design_surface_schema`. (See also figures G.1 to G.3 and G.5 to G.17)

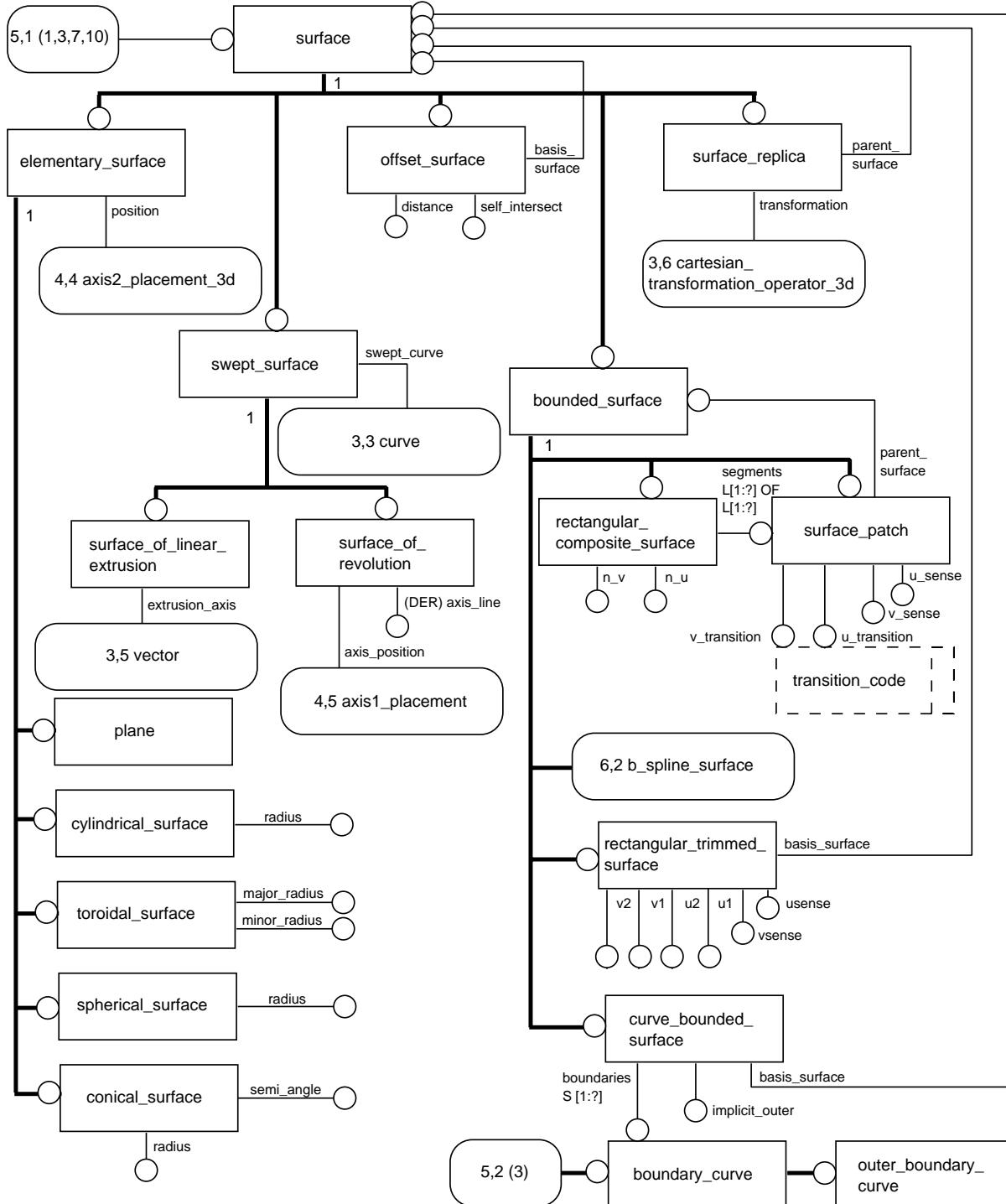


Figure G.5 – Graphical notation of the major aspects of surfaces in the mechanical\_surface\_schema. (See also figures G.1 to G.4 and G.6 to G.17)

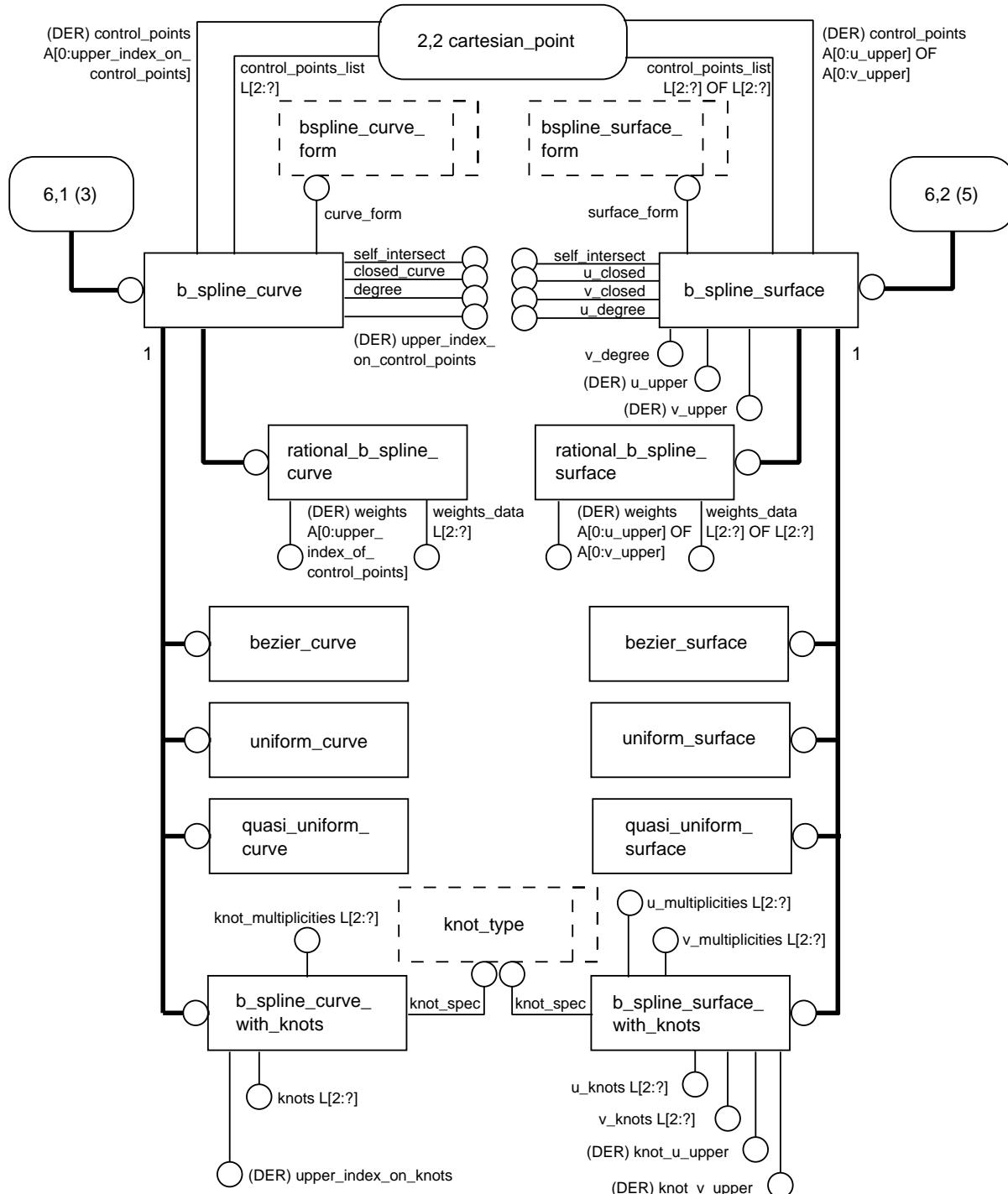


Figure G.6 – Graphical notation of the major aspects of B-spline geometry in the mechanical\_design\_surface\_schema. (See also figures G.1 to G.5 and G.7 to G.17)

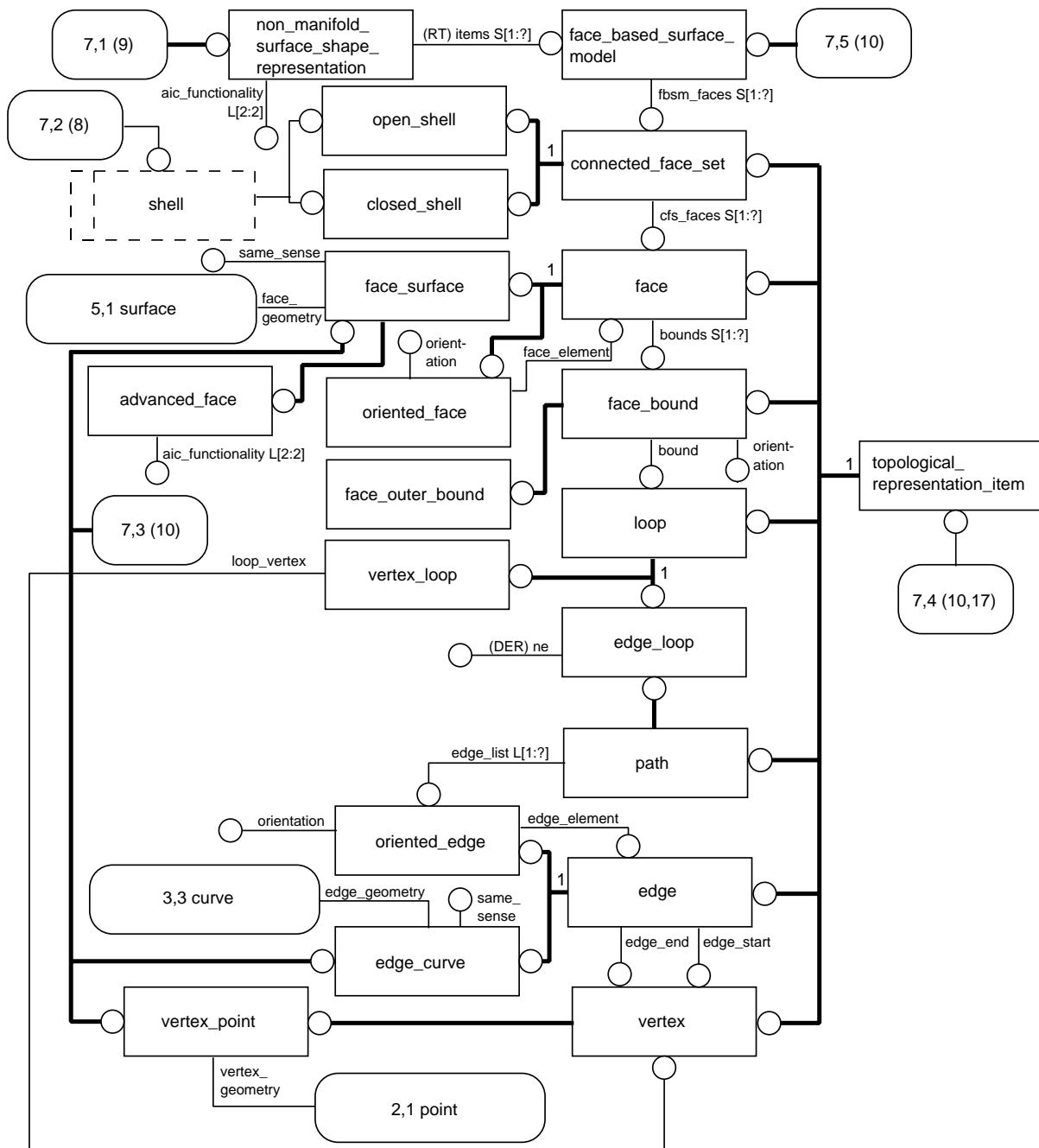


Figure G.7 – Graphical notation of the major aspects of topology in the mechanical\_design\_surface\_schema. (See also figures G.1 to G.6 and G.8 to G.17)

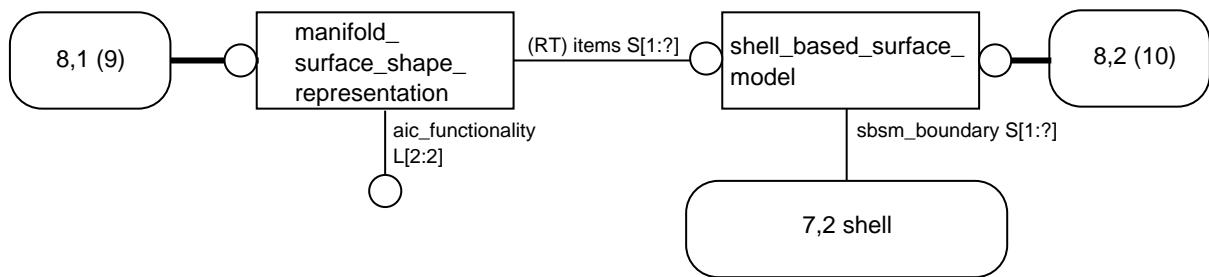


Figure G.8 – Graphical notation of the major aspects of topology (continued) in the mechanical\_design\_surface\_schema. (See also figures G.1 to G.7 and G.9 to G.17)

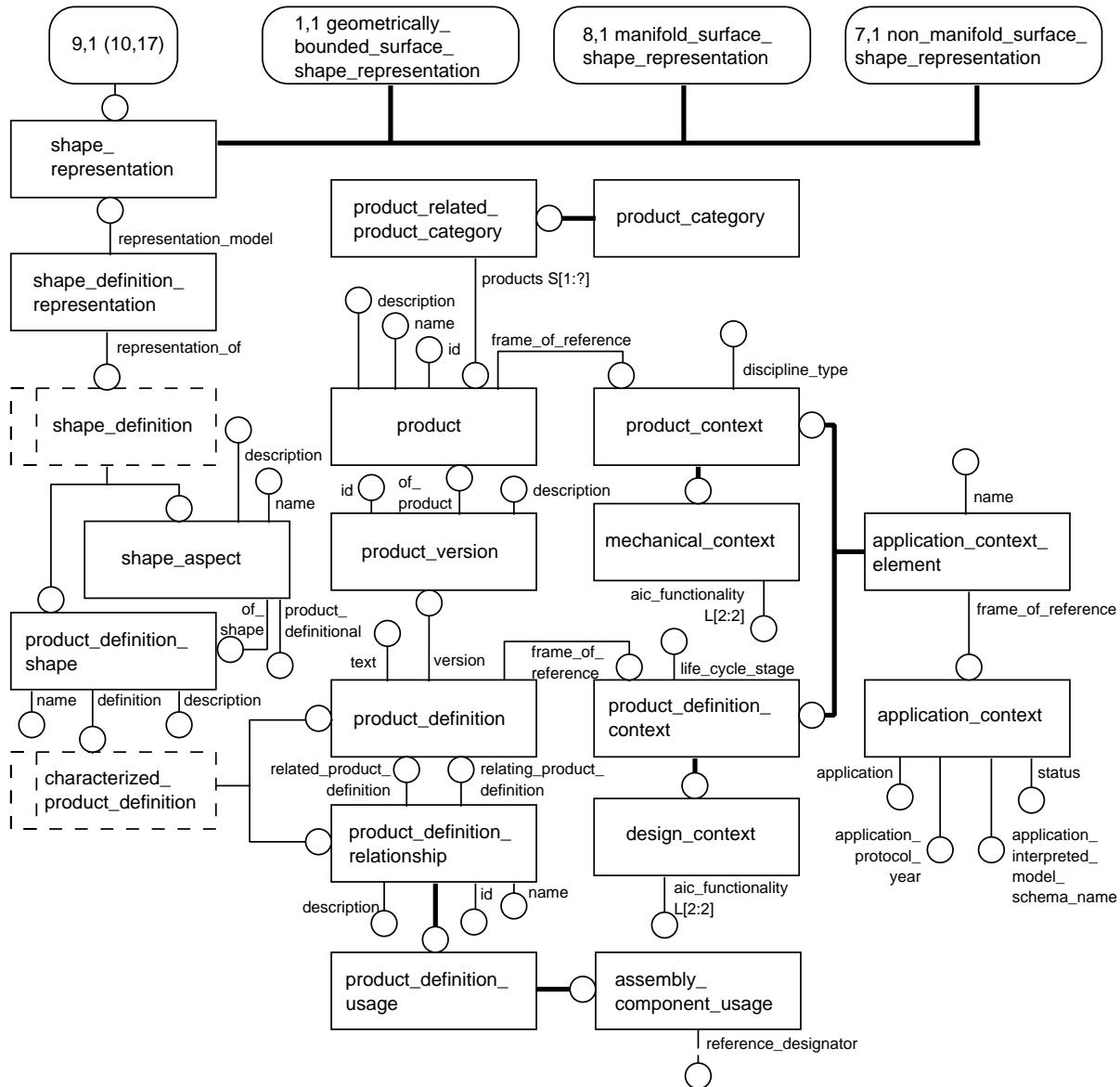


Figure G.9 – Graphical notation of the major aspects of product structure in the mechanical\_design\_surface\_schema. (See also figures G.1 to G.8 and G.10 to G.17)

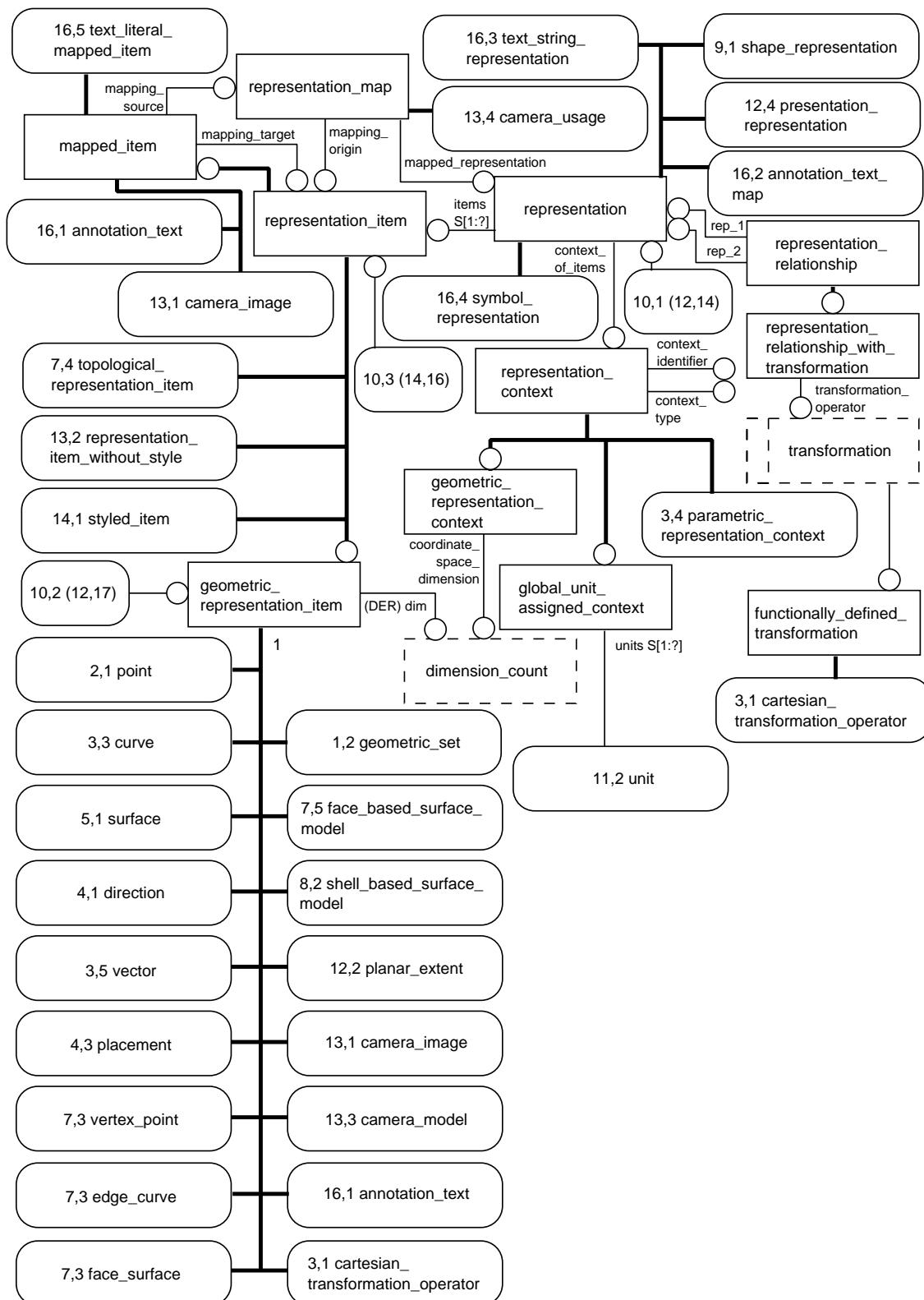


Figure G.10 – Graphical notation of the major aspects of product structure (continued) in the mechanical\_design\_surface\_schema. (See also figures G.1 to G.9 and G.11 to G.17)

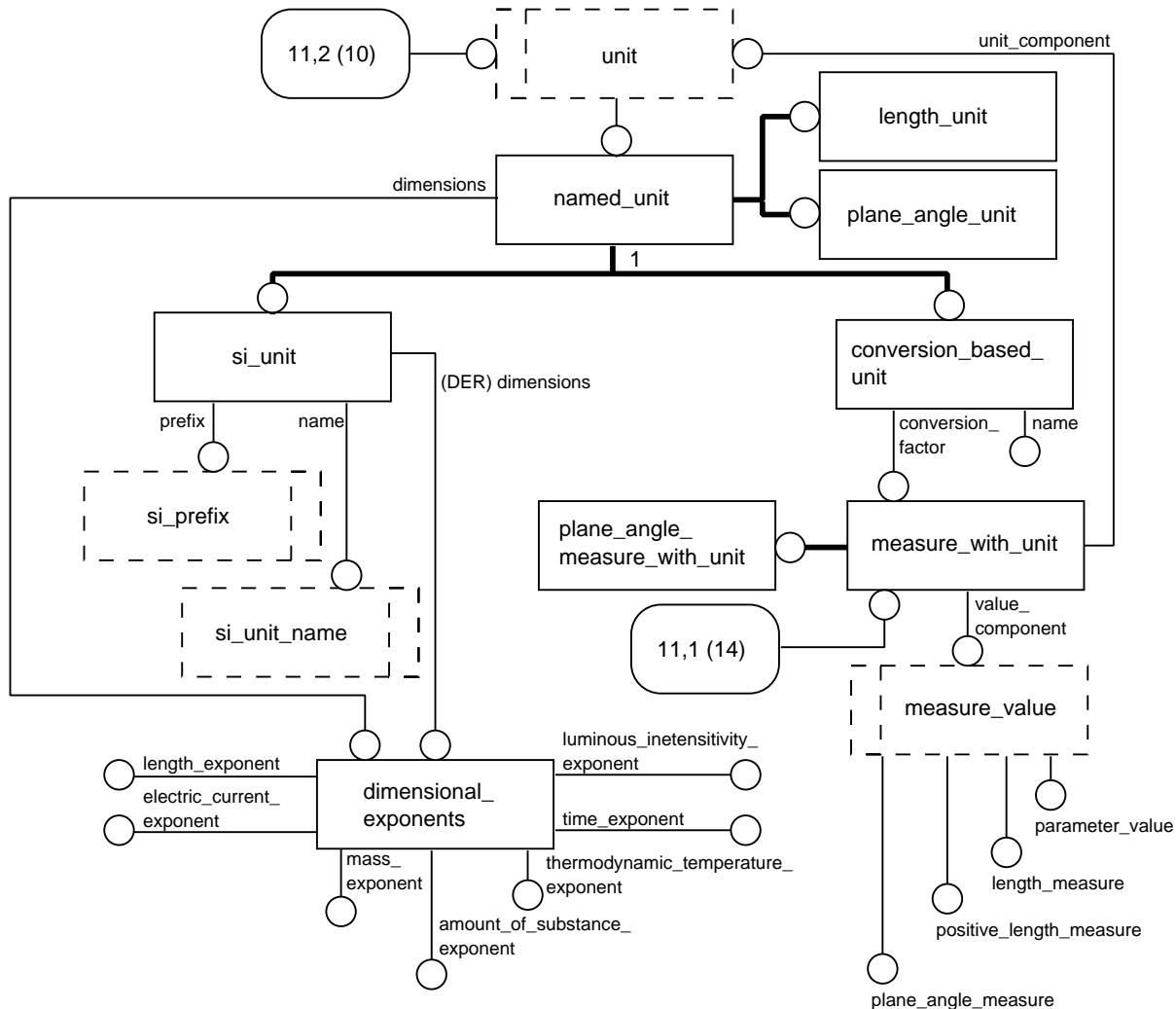


Figure G.11 – Graphical notation of the major aspects of units representation in the mechanical\_design\_surface\_schema. (See also figures G.1 to G.10 and G.12 to G.17)

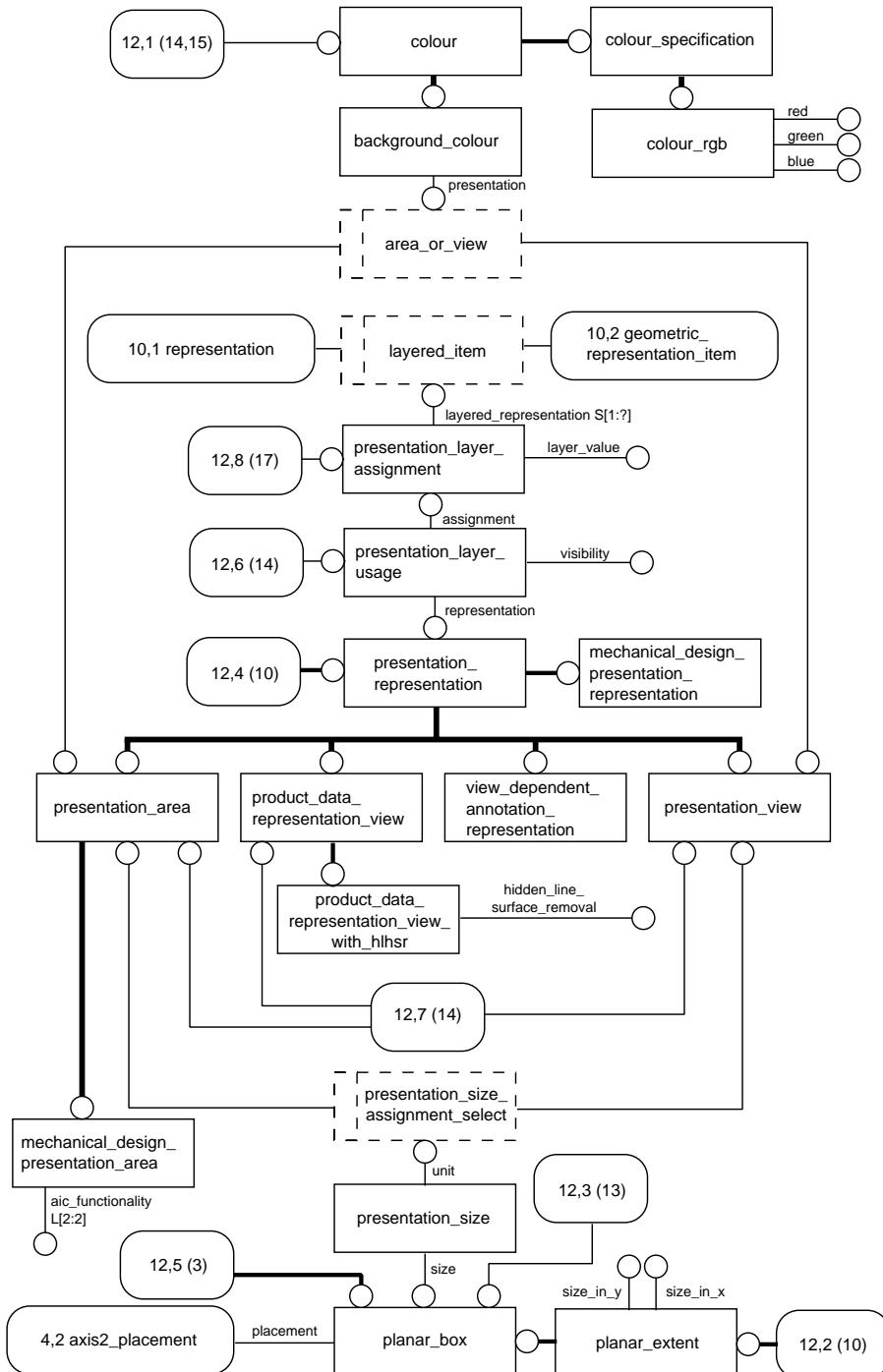


Figure G.12 – Graphical notation of the major aspects of visual presentation in the `mechanical_design_surface_schema`. (See also figures G.1 to G.11 and G.13 to G.17)

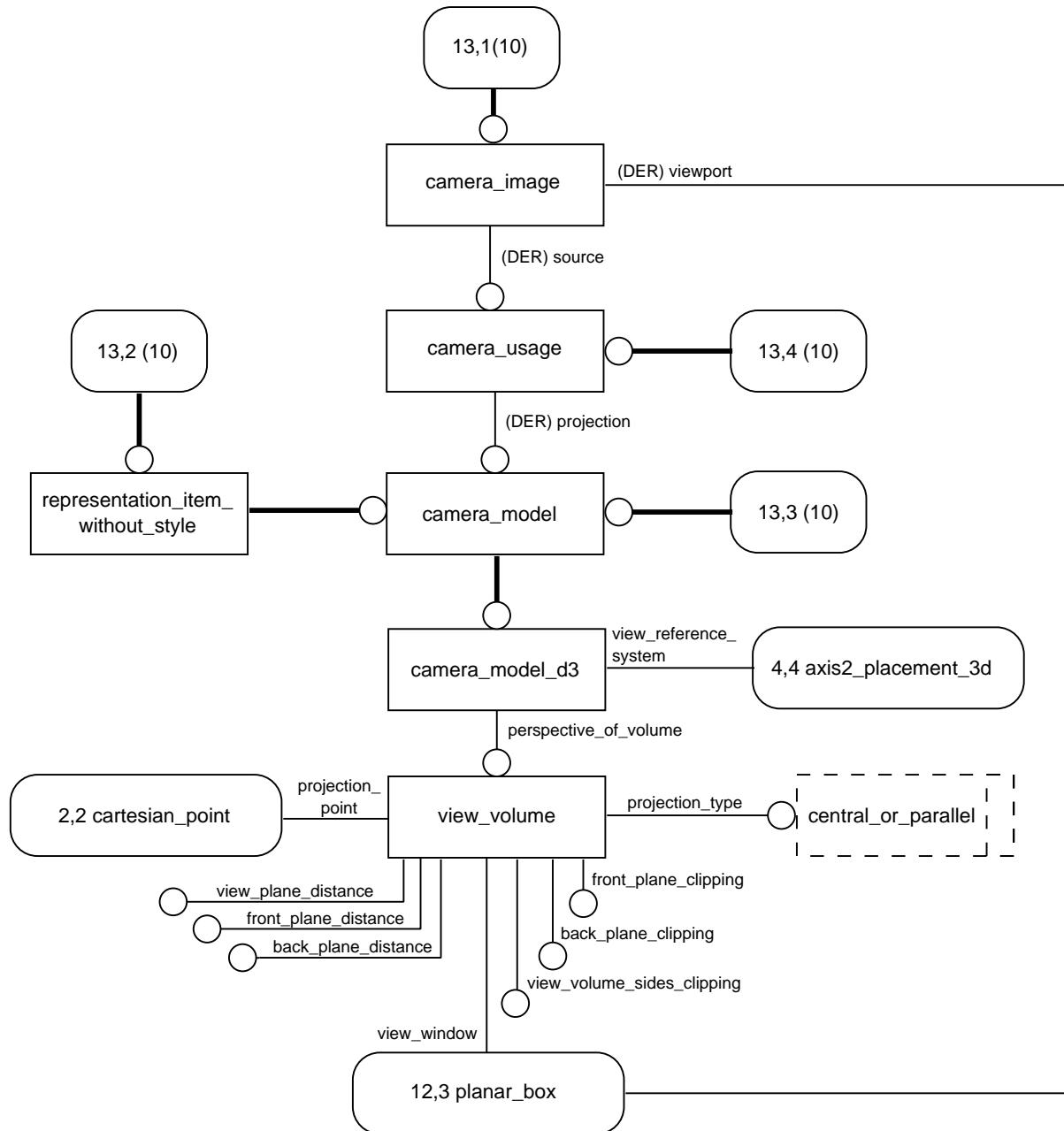


Figure G.13 – Graphical notation of the major aspects of visual presentation (continued) in the mechanical\_design\_surface\_schema. (See also figures G.1 to G.12 and G.14 to G.17)

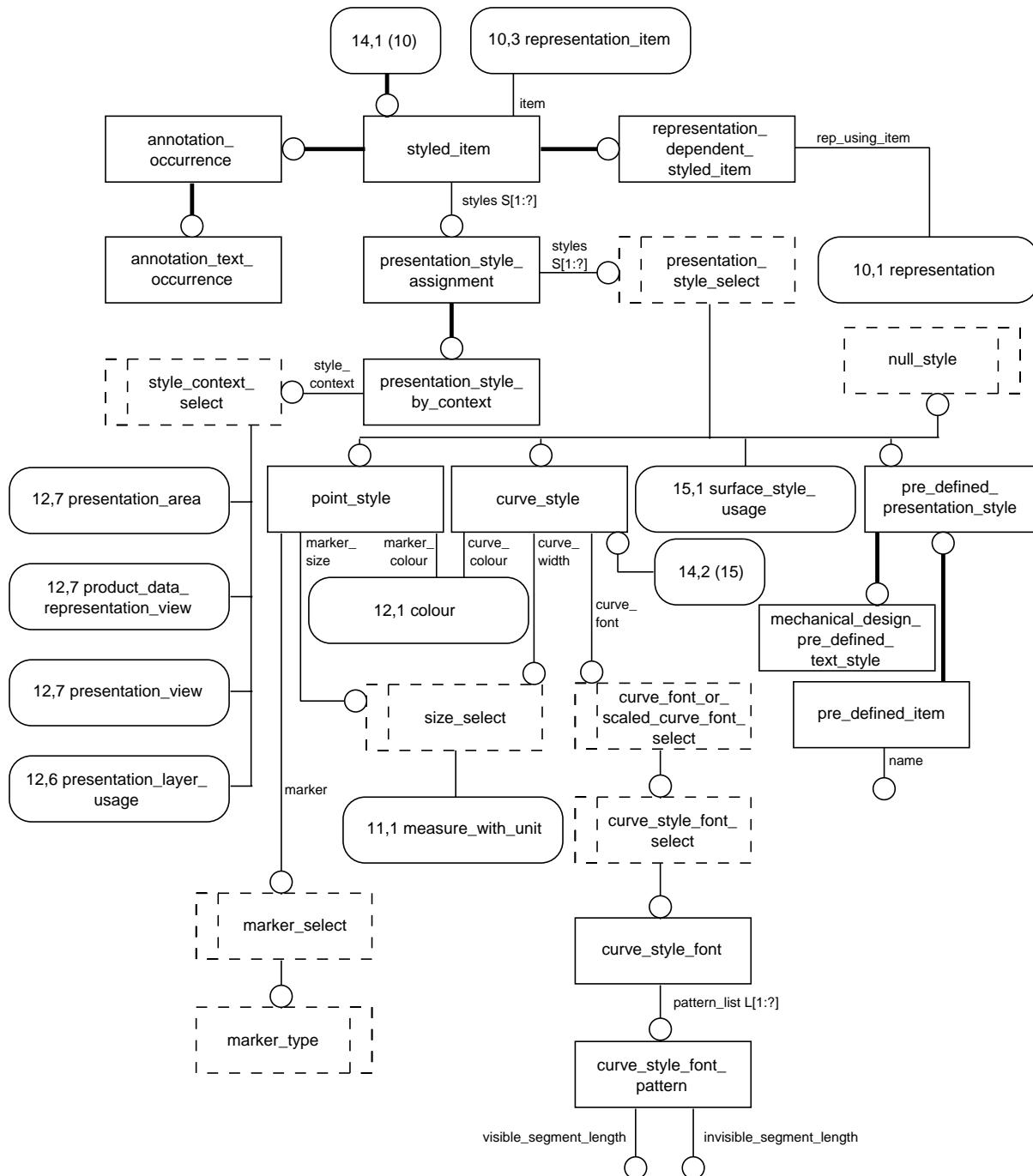


Figure G.14 – Graphical notation of the major aspects of visual presentation (continued) in the mechanical\_design\_surface\_schema. (See also figures G.1 to G.13 and G.15 to G.17)

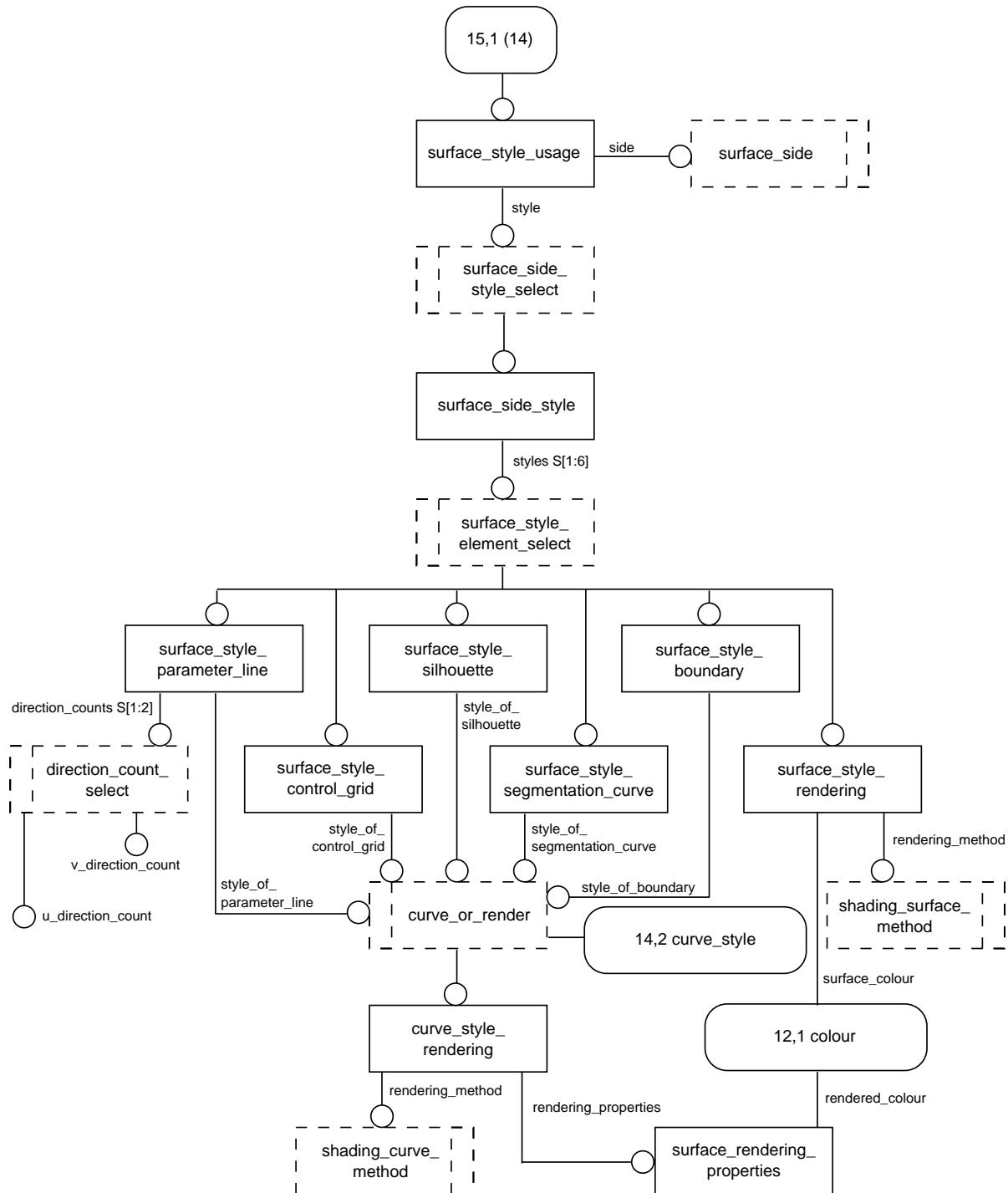


Figure G.15 – Graphical notation of the major aspects of visual presentation (continued) in the mechanical\_design\_surface\_schema. (See also figures G.1 to G.14 and G.16 to G.17)

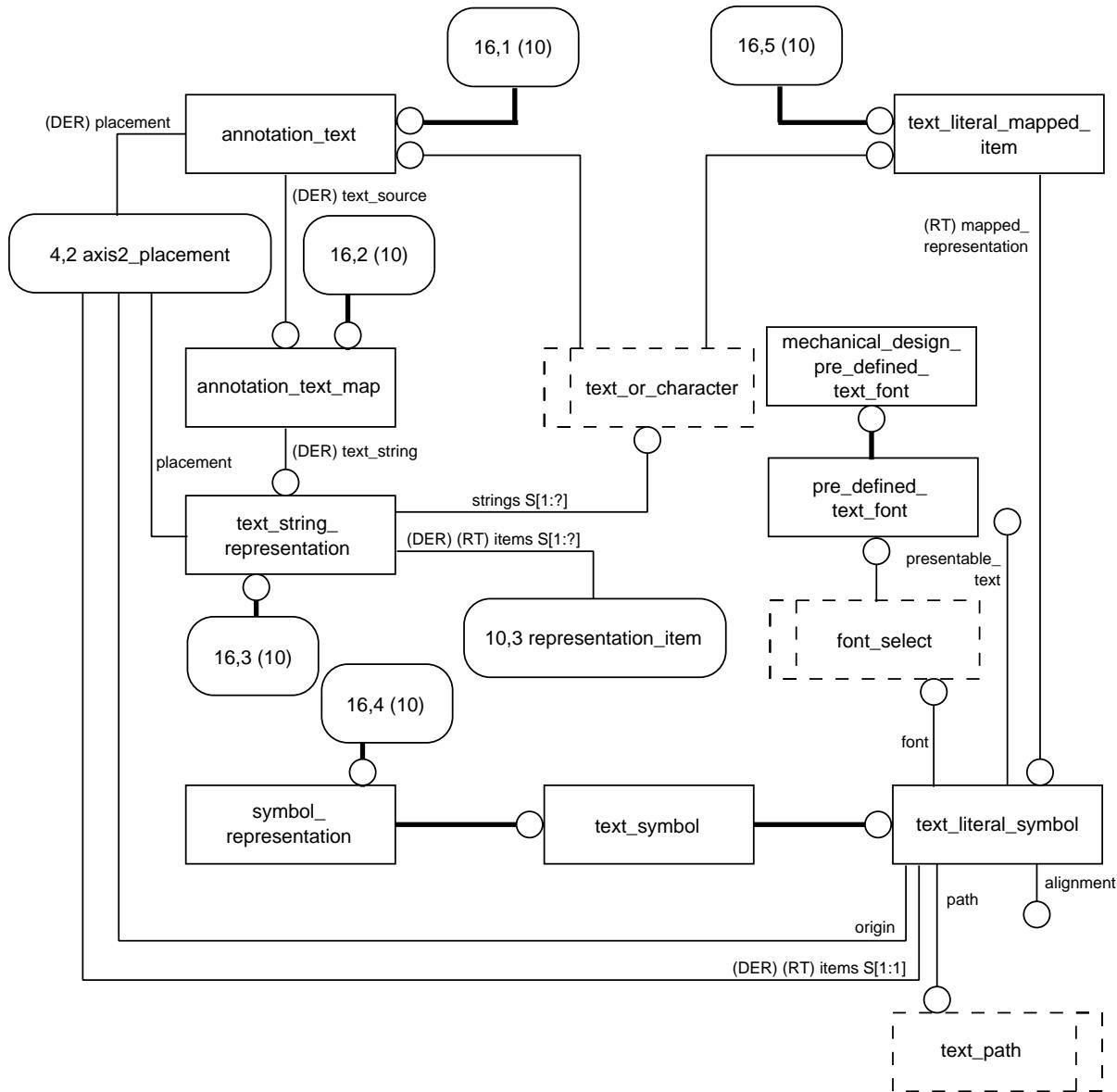


Figure G.16 – Graphical notation of the major aspects of visual presentation (continued) in the `mechanical_design_surface_schema`. (See also figures G.1 to G.15 and G.17)

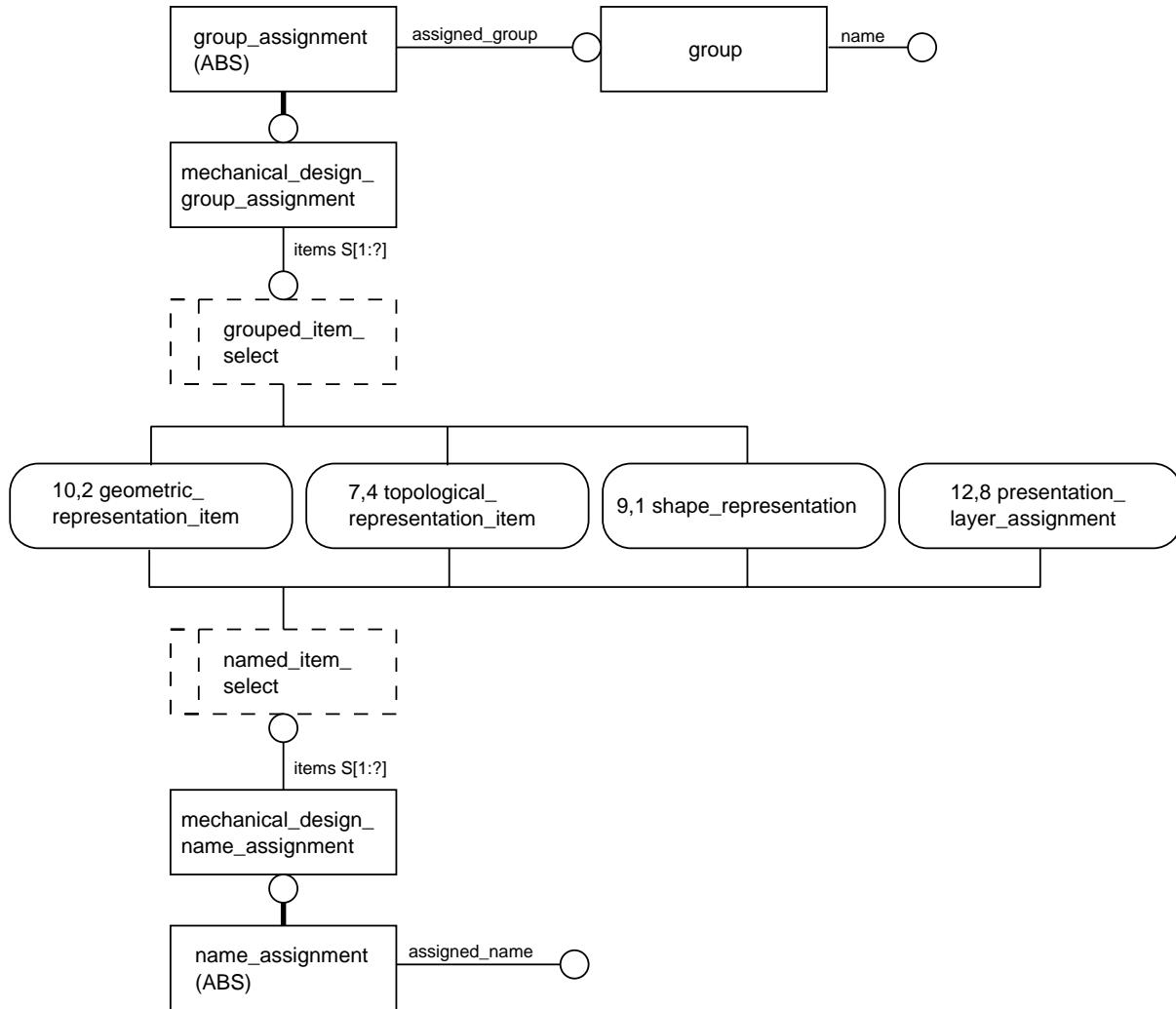


Figure G.17 – Graphical notation of the major aspects of group and name assignment in the mechanical\_design\_surface\_schema. (See also figures 1 to 16)

**Annex H**  
(informative)

**AIM EXPRESS listing**

This annex provides a listing of the table of short names and a listing of the *EXPRESS* specified in the AIM of this part of ISO 10303. No text or annotation is included. This annex is provided only in computer-interpretable form.

NOTE – The information provided on this diskette is informative; the normative text is that contained in the body of this part of ISO 10303.

**Annex J**  
(informative)

**Bibliography**

The IDEF0 activity models are in their format based on the following document:

**Annex K**  
(informative)

**Application protocol usage guide**

## Index

<i>d</i> -manifold with boundary .....	9
2-manifold .....	11
3D_projection	
definition .....	27
mapping table .....	86
3D_projection to Screen_image .....	42
3D_projection to Surface_model .....	42
<b>A</b>	
Accuracy of numbers .....	233
advanced_face	
mapping table .....	62, 72
AIM long listing .....	125
alternative_surface_shape_representations	
AIM EXPRESS short listing .....	110
Annotation_text	
definition .....	27
mapping table .....	88
Annotation_text to Screen_image .....	42
Annotation_text to Transformation .....	42
annotation_text_occurrence	
mapping table .....	88
application .....	7
application activity model (AAM) .....	7
application context .....	7
application interpreted construct (AIC) .....	11
application interpreted model (AIM) .....	7
application object .....	11
application protocol (AP) .....	7
application reference model (ARM) .....	7
arcwise connected .....	8
assembly .....	7
ARM .....	249
definition .....	27
mapping table .....	80
Assembly to Assembly .....	42
Assembly to Layer .....	42
Assembly to Part .....	42
Assembly to Product .....	42
Assembly to Transformation .....	42
axi-symmetric .....	8
Axis_placement	
definition .....	27
mapping table .....	51, 60, 70
<b>B</b>	
B_spline_curve	
definition .....	27
mapping table .....	51, 60, 70
b_spline_curve_with_knots	
mapping table .....	51, 60, 70

B_spline_surface	
definition .....	28
mapping table .....	51, 60, 70
b_spline_surface_with_knots	
mapping table .....	51, 60, 70
boundary .....	8
boundary representation (B-rep) .....	11
Bounded_surface	
definition .....	28
Bounded_curve	
ARM .....	259, 265
definition .....	28
mapping table .....	51, 60, 70
bounded_surface	
ARM .....	262, 265
mapping table .....	51, 60, 70
bounds .....	8
<b>C</b>	
camera_model_d3	
mapping table .....	86
cartesian_point	
definition .....	28
mapping table .....	51, 60, 70
cartesian_transformation_operator	
mapping table .....	58
cartesian_transformation_operator_3d	
mapping table .....	69, 79
circle	
definition .....	28
mapping table .....	51, 60, 70
closed curve .....	8
closed surface .....	8
Closed_shell	
definition .....	29
mapping table .....	60, 70
colour_rgb	
mapping table .....	91, 97, 99
completion of a topological entity .....	8
component .....	7
composite_curve	
definition .....	29
mapping table .....	51
Composite_curve_on_surface	
ARM .....	260
definition .....	29
mapping table .....	52
Composite_curve_on_surface to Intersection_curve .....	43
Composite_curve_on_surface to Parametric_curve .....	43
Composite_curve_on_surface to Seam_curve .....	43
conformance assessment process .....	8
conformance class .....	8

conformance testing .....	7
Conical_surface	
definition .....	29
mapping table .....	53, 60, 70
connected .....	8
connected component .....	8
connected_face_set	
mapping table .....	63, 67, 73, 77
constructive solid geometry (CSG) .....	11
context .....	7
coordinate space .....	10
curve .....	8
ARM .....	258
definition .....	29
mapping table .....	53, 61, 71
Curve to Curve_appearance .....	43
Curve to Curve_replica .....	43
Curve to Edge .....	43
Curve to Geometrically_bounded_surface_model .....	43
Curve_2d	
definition .....	29
mapping table .....	53, 61, 71
Curve_2D to Parametric_curve. ....	43
Curve_3D	
definition .....	29
mapping table .....	53, 61, 71
Curve_3D to Intersection_curve .....	43
Curve_3D to Seam_curve .....	44
Curve_appearance	
definition .....	30
mapping table .....	91
Curve_appearance to Edge .....	44
Curve_bounded_surface	
definition .....	30
mapping table .....	54
Curve_on_surface	
ARM .....	259, 265
definition .....	30
mapping table .....	54, 62, 72
Curve_replica	
definition .....	30
mapping table .....	54–55, 62, 64, 72, 74
curve_style	
mapping table .....	91
curve_style_font	
mapping table .....	91
cycle .....	8
cylindrical_surface	
definition .....	31
mapping table .....	54, 62, 72

## ISO/CD 10303-205

data .....	7
data exchange .....	7
degenerate_pcurve	
mapping table .....	54, 62, 72
Degenerated_parametric_curve	
definition .....	31
mapping table .....	54, 62, 72
dependent_instantiation_of_geometry	
AIM EXPRESS short listing .....	110
dependent_instantiation_of_mapped_item	
AIM EXPRESS short listing .....	111
dependent_instantiation_of_mechanical_design_presentation_representation	
AIM EXPRESS short listing .....	111
dependent_instantiation_of_named_unit	
AIM EXPRESS short listing .....	112
dependent_instantiation_of_product_context	
AIM EXPRESS short listing .....	112
dependent_instantiation_of_product_definition	
AIM EXPRESS short listing .....	112
dependent_instantiation_of_product_definition_context	
AIM EXPRESS short listing .....	113
dependent_instantiation_of_product_definition_relationship	
AIM EXPRESS short listing .....	113
dependent_instantiation_of_product_related_product_category	
AIM EXPRESS short listing .....	114
dependent_instantiation_of_product_version	
AIM EXPRESS short listing .....	114
dependent_instantiation_of_shape_representation	
AIM EXPRESS short listing .....	115
dependent_instantiation_of_styled_item	
AIM EXPRESS short listing .....	115
dependent_instantiation_of_topology	
AIM EXPRESS short listing .....	115
dimensionality .....	8
domain .....	9
<b>E</b>	
Edge	
definition .....	31
mapping table .....	62, 72
Edge to Loop .....	44
Edge to Vertex .....	44
elementary_surface	
definition .....	31
mapping table .....	54, 62, 72
Ellipse	
definition .....	31
mapping table .....	54, 62, 72
Entities	
short form .....	105
Entity definitions	
short form .....	105

Euler equations .....	9
extent .....	9
externally defined .....	10
<b>F</b>	
Face	
ARM .....	264
definition .....	31
mapping table .....	62, 72
Face to Face_set .....	44
Face to Loop .....	44
Face to Surface .....	44
Face to Surface_appearance .....	44
face_based_surface_model	
mapping table .....	75
Face_set	
definition .....	31
mapping table .....	63, 73
Face_set to Manifold_surface_model .....	44
Face_set to Non_manifold_surface_model .....	45
Face_set to Presentation_appearance .....	45
File implementation .....	233
finite .....	9
finite element analysis (FEA) .....	11
functional data group (FDG) .....	11
Functional data group 1	
AIM .....	15
Functional data group 2	
AIM .....	15
Functional data group 3	
AIM .....	16
<b>G</b>	
generic resource .....	7
genus of a graph .....	9
genus of a surface .....	9
geometric coordinate system .....	9
Geometric_element	
definition .....	32
mapping table .....	55, 64, 74
Geometric_element to Group .....	45
Geometric_element to Layer .....	45
geometric_representation_context	
AIM EXPRESS short listing .....	106
rule for	
short form .....	116
geometric_representation_item	
AIM EXPRESS short listing .....	106
mapping table .....	55, 64, 74
rule for	
short form .....	110, 118
geometric_set	
mapping table .....	54

## ISO/CD 10303-205

geometric_set_replica	57
mapping table .....	
geometrical_representation_item	
rule for	
short form .....	120
geometrically bounded surface model .....	11
geometrically founded .....	10
geometrically related .....	10
Geometrically_bounded_surface_model	
ARM .....	257
definition .....	32
mapping table .....	54
Geometrically_bounded_surface_model to Point .....	45
Geometrically_bounded_surface_model to Surface .....	45
geometrically_bounded_surface_model_shape	
mapping table .....	51
UoF definition .....	17
geometrically_bounded_surface_shape_representation	
mapping table .....	57
Geometry	
ARM .....	252
Geometry_replica	
ARM .....	253
definition .....	32
mapping table .....	55, 64, 74
Geometry_replica to Transformation .....	45
global_unit_assigned_context	
AIM EXPRESS short listing .....	106
mapping table .....	83
rule for	
short form .....	116
global_units_in_geometric_representation_context	
AIM EXPRESS short listing .....	116
global_units_required	
AIM EXPRESS short listing .....	116
graph .....	9
group	
definition .....	32
mapping table .....	59
Group to Surface_model .....	45
Group to Topological_element .....	45
grouped_item_select	
AIM EXPRESS short listing .....	104
grouping	
mapping table .....	59
UoF definition .....	19
<b>H</b>	
handle .....	9
homeomorphic .....	9
Hyperbola	
definition .....	32

mapping table .....	55, 64, 74
<b>I</b>	
illegal_complex_named_units	
AIM EXPRESS short listing .....	117
implementation method .....	7
Implementation method specific requirements .....	233
information .....	7
Information requirements clause .....	14
initial graphics exchange specification (IGES) .....	11
inside .....	9
integrated resource (IR) .....	7
interior .....	9
interpretation .....	7
Intersection_curve	
ARM .....	261
definition .....	33
mapping table .....	55, 64, 74
Intersection_curve to Surface .....	46
<b>L</b>	
Layer	
ARM .....	256
definition .....	33
mapping table .....	92
Layer to Layer .....	46
Layer to Part .....	46
Layer to Presentation_appearance .....	46
Layer to Product .....	46
Layer to Surface_model .....	46
Layer to Topological_element .....	46
line	
definition .....	33
mapping table .....	56, 58, 65, 69, 75, 79
list .....	9
Loop	
definition .....	33
mapping table .....	65, 75
Loop to Vertex .....	46
<b>M</b>	
manifold surface model .....	11
Manifold_surface_model	
definition .....	33
mapping table .....	65
manifold_surface_model_shape	
mapping table .....	60
UoF definition .....	20
manifold_surface_shape_representation	
mapping table .....	67
mapped_item	
AIM EXPRESS short listing .....	107
mapping table .....	67, 77

## ISO/CD 10303-205

rule for	
short form .....	111
Mapping to physical file .....	233
marker_type	
mapping table .....	97
mechanical_design_group_assignment	
AIM EXPRESS short listing .....	105
mechanical_design_name_assignment	
AIM EXPRESS short listing .....	105
mapping table .....	55, 57, 64, 67–68, 74, 77–78, 92
mechanical_design_presentation_area	
mapping table .....	99
mechanical_design_presentation_representation	
rule for	
short form .....	111
model .....	7
<b>N</b>	
named_item_select	
AIM EXPRESS short listing .....	105
named_unit	
AIM EXPRESS short listing .....	107
rule for	
short form .....	112, 117
NIAM-guide .....	266
no_complex_geometric_representation_item	
AIM EXPRESS short listing .....	118
non-manifold surface model .....	11
Non_manifold_surface_model	
definition .....	33
mapping table .....	75
non_manifold_surface_model_shape	
mapping table .....	70
UoF definition .....	22
non_manifold_surface_shape_representation	
mapping table .....	77
numerical control (NC) .....	11
<b>O</b>	
Offset_curve	
definition .....	34
mapping table .....	56, 65, 76
offset_curve_3d	
mapping table .....	56, 65, 76
offset_surface	
definition .....	34
mapping table .....	56, 66, 76
open_curve .....	9
open_surface .....	9
open_shell	
definition .....	34
mapping table .....	66, 76
orientable .....	9

overlap .....	9
<b>P</b>	
Parabola	
definition .....	34
mapping table .....	56, 66, 76
parameter range .....	9
parameter space .....	9
Parametric_curve	
ARM .....	261
definition .....	34
mapping table .....	56, 66, 76
Parametric_curve to Seam_curve .....	46
Parametric_curve to Surface .....	47
Part	
ARM .....	249
definition .....	34
mapping table .....	81
Part to Product .....	47
Part to Shape_representation .....	47
Part to Transformation .....	47
pcurve	
mapping table .....	56, 66, 76
Physical file	
accuracy of numbers .....	233
physical state variable .....	10
PICS .....	7
picture .....	10
PIXIT .....	7
placement	
mapping table .....	51, 60, 70
placement coordinate system .....	9
plane	
definition .....	35
mapping table .....	56, 66, 76
Point	
ARM .....	257
definition .....	35
mapping table .....	56, 66, 76
Point to Point_appearance .....	47
Point to Vertex .....	47
Point_appearance	
definition .....	35
mapping table .....	97
point_on_curve	
definition .....	35
mapping table .....	56, 66, 76
point_on_surface	
definition .....	36
mapping table .....	56, 66, 76
point_style	
mapping table .....	97

## ISO/CD 10303-205

Polyline	
definition .....	36
mapping table .....	56
predefined items .....	10
presentation .....	10
ARM	
attributes .....	255
presentation information .....	10
Presentation_appearance	
definition .....	36
mapping table .....	98
Presentation_appearance to Surface_model .....	47
presentation_area	
AIM EXPRESS short listing .....	107
rule for	
short form .....	119
presentation_area_mechanical_design	
AIM EXPRESS short listing .....	119
presentation_layer_assignment	
mapping table .....	92
presentation_style_assignment	
mapping table .....	98
product .....	8
ARM .....	249
definition .....	36
mapping table .....	80, 82–83
product data .....	8
product shape .....	10
product_context	
AIM EXPRESS short listing .....	107
rule for	
short form .....	112, 119
product_context_mechanical	
AIM EXPRESS short listing .....	119
product_definition	
AIM EXPRESS short listing .....	108
rule for	
short form .....	112
product_definition_context	
AIM EXPRESS short listing .....	108
rule for	
short form .....	113, 120
product_definition_context_design	
AIM EXPRESS short listing .....	120
product_definition_relationship	
AIM EXPRESS short listing .....	108
mapping table .....	80
rule for	
short form .....	113
product_definition_shape	
mapping table .....	81
product_related_product_category	

AIM EXPRESS short listing .....	108
rule for	
short form .....	114
product_structure	
mapping table .....	80
UoF definition .....	25
product_version	
AIM EXPRESS short listing .....	109
mapping table .....	82–83
rule for	
short form .....	114
<b>R</b>	
realistic presentation of properties .....	10
Rectangular_composite_surface	
definition .....	37
mapping table .....	56
rectangular_trimmed_surface	
definition .....	37
mapping table .....	56
Reference to AIM schema .....	233
representation	
mapping table .....	80, 82
resource construct .....	8
Rules	
AIM .....	110
ARM-AIM mapping .....	102, 113
<b>S</b>	
Screen_image	
definition .....	37
mapping table .....	99
Seam_curve	
ARM .....	260
definition .....	37
mapping table .....	56, 66, 76
self-intersect .....	9
self-loop .....	9
set .....	9
shape_aspect	
mapping table .....	84
shape_representation	
AIM EXPRESS short listing .....	109
definition .....	37
mapping table .....	83
rule for	
short form .....	110, 115
Shell	
definition .....	38
mapping table .....	67, 77
shell_based_surface_model	
mapping table .....	65, 68, 78

space dimensionality .....	10
Spherical_surface	
definition .....	38
mapping table .....	56, 67, 77
staircase function .....	10
standard d'échange et de transfert (SET) .....	11
structure .....	8
styled_item	
AIM EXPRESS short listing .....	109
rule for	
short form .....	115
surface .....	10
ARM .....	262
definition .....	38
mapping table .....	57, 67, 77
Surface to Surface_appearance .....	47
Surface to Surface_replica .....	47
Surface_appearance	
definition .....	38
mapping table .....	99
surface_basic_visual_presentation	
mapping table .....	86
UoF definition .....	26
surface_curve	
mapping table .....	62, 72
Surface_model	
ARM .....	250
definition .....	39
mapping table .....	57, 67, 77
Surface_model to Surface_model_replica .....	48
Surface_model_replica	
ARM .....	251
definition .....	39
mapping table .....	57, 67, 77
Surface_model_replica to Transformation .....	48
surface_of_linear_extrusion	
definition .....	40
mapping table .....	58, 68, 78
surface_of_revolution	
definition .....	40
mapping table .....	58, 68, 78
surface_replica	
definition .....	40
mapping table .....	58, 68, 78
surface_style_control_grid	
mapping table .....	100
surface_style_usage	
mapping table .....	99
swept_surface	
definition .....	40
mapping table .....	58, 68, 78
symbol .....	10

symbolic presentation of properties .....	10
synthetic camera model .....	10
<b>T</b>	
test purpose .....	8
three_dimensional_geometry_mainly	
AIM EXPRESS short listing .....	120
topological sense .....	10
Topological_element	
definition .....	40
mapping table .....	68, 78
topological_representation_item	
AIM EXPRESS short listing .....	109
mapping table .....	68, 78
rule for	
short form .....	115
Topology	
ARM .....	254
Topology_surface_model	
ARM .....	263
definition .....	41
mapping table .....	68, 78
Toroidal_surface	
definition .....	41
mapping table .....	58, 68, 78
transformation	
definition .....	41
mapping table .....	58, 69, 79, 84
trimmed_curve	
definition .....	41
mapping table .....	58
Types	
short form .....	104
<b>U</b>	
Unbounded_curve	
ARM .....	258
definition .....	41
mapping table .....	58, 69, 79
unit of functionality (UoF) .....	8
Units of functionality	
geometrically_boundedsurface_model .....	17
grouping .....	19
manifold_surface_model .....	20
non_manifold_surface_model .....	22
product_structure .....	25
surface_basic_visual_presentation .....	26
<b>V</b>	
Verband der Automobilindustrie - Flaechenschnittstelle (VDAFS) .....	11
Verband der Automobilindustrie - IGES subset (VDAIS) .....	12
Vertex	
definition .....	41

## **ISO/CD 10303-205**

mapping table .....	69, 79
vertex_point	
mapping table .....	69, 79
visualisation .....	10